# Inferring Sensitive Information from Seemingly Innocuous Smartphone Data

Anthony Quattrone

Submitted in total fulfillment of the requirements of the degree of

**Doctor of Philosophy**

Department of Computing and Information Systems
The University of Melbourne, Australia

December 2016

# Abstract

Smartphones have become ubiquitous and provide considerable benefits for users. Personal data considered both sensitive and non-sensitive is commonly stored on smartphones. It has been established that the use of smartphones can lead to users enjoying less privacy when a data leak occurs. In this thesis, we aim to determine if smartphone stored non-sensitive data can be analyzed to derive sensitive information. We also aim to develop methods for protecting smartphone users from privacy attacks. While privacy research is an active area, more work is needed to determine the types of inferences that can be made about a person and how accurate a profile is derivable from smartphone data.

We demonstrate how straightforward it is for third party app developers to embed code in inconspicuous apps to capture and mine data. Our studies show that there are a large number of apps found in popular app stores commonly requesting special app permissions in order to gain access to sensitive data. In most cases, we could not find any functional benefits in exchange for accessing the sensitive data. Additional data not local to the device but rather stored on a social networking service can also be extracted and unnoticed via mobile apps provided the user has social networking apps installed. Current research shows that users do not easily comprehend the implications of granting apps access to sensitive permissions. Apps can transfer captured data to cloud services easily via high-speed wireless networks. This is difficult for users to detect since mobile platforms do not provide alerts for when this occurs. With access to sensitive mobile data, we developed and performed a number of case studies to learn details about an individual. Modern smartphones are capable of sending continuous location updates to services providing a near real-time proxy of where a user is located. We were able to determine where a user was traveling simply by analyzing the point of interest results from continuous location queries sent to a location-based service without referencing user location data. With continuous location data, we were able to determine

the personal encounters between individuals and relationships in realtime. We also found that even diagnostic data commonly used to debug apps that appears to be anonymous is useful in identifying an individual. Mobile devices disclose the indoor location information of the user indirectly via the wireless signals emitted by the device. We were able to locate users indoors by analyzing Bluetooth beacons within a 1m accuracy by using a localization scheme we developed. With the understanding that mobile data is sometimes needed by apps to provide functionality, we developed a system called PrivacyPalisade designed to protect users from revealing sensitive information. The system detects when apps are requesting uncharacteristic app permissions based on the app's category.

Overall we found that mobile smartphones store seemingly non-sensitive data that can reveal sensitive information upon further inspection and that this information is easily accessible. In turn, an analyst can use this data to build a detailed individual profile of private and sensitive information. With the growing number of users expressing privacy concerns, techniques to better protect privacy are needed to allow manufacturers to meet their users' privacy requirements. Our proposed protection methods demonstrated in PrivacyPalisade can be adopted to make smartphone platforms more privacy aware.

**Thesis Contributions:** In this thesis, we show how sensitive information can be inferred from seemingly innocuous data and propose a protection system by performing the following:

- Provide a comprehensive literature review of the current state of privacy research and how it relates to the use of smartphones.
- Demonstrate an inference attack on trajectory privacy by reconstructing a route using only query results.
- Develop an algorithm that combines range-based and range-free localization schemes to perform indoor localization using Bluetooth with accuracies of up to 1m.
- Analyze diagnostic data commonly sent by smartphones and used it to identify users in a dataset with accuracies of up to 97%.
- Develop an algorithm to infer potential encounters of smartphone users in realtime by proposing the use of a constraint nearest neighbor (c-NN) spatial query.
- Develop and demonstrate PrivacyPalisade, a system developed for Android with the aim of protecting against privacy attacks.

# Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD except where indicated in the preface,
2. due acknowledgment has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies, and appendices.

Anthony Quattrone, December 2016

# Preface

The research conducted throughout the completion of this thesis has led to publications contributing to the field. I am the primary author of these papers:

- **Chapter 3** is based on the work **Tell Me What You Want and I Will Tell Others Where You Have Been**, Anthony Quattrone, Elham Naghizade, Lars Kulik, Egemen Tanin - CIKM '14 Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. Please note that a preliminary study on the use of Voronoi diagrams was completed in an earlier honors research project.

- **Chapter 4** is based on the work **Combining Range-Based and Range-Free Methods: A Unified Approach for Localization**, Anthony Quattrone, Lars Kulik, Egemen Tanin - SIGSPATIAL'15 Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems.

- **Chapter 5** is based on the work **Mining City-Wide Encounters in Real-Time**, Anthony Quattrone, Lars Kulik, Egemen Tanin - SIGSPATIAL'16 - Proceedings of the 24rd SIGSPATIAL International Conference on Advances in Geographic Information Systems.

- **Chapter 6** is based on the work **Is This You?: Identifying a Mobile User Using Only Diagnostic Features**, Anthony Quattrone, Tanusri Bhattacharya, Lars Kulik, Egemen Tanin, James Bailey - MUM '14 Proceedings of the 13th International Conference on Mobile and Ubiquitous Multimedia.

- **Chapter 7** is based on the work **PrivacyPalisade: Evaluating App Permissions and Building Privacy into Smartphones**, Anthony Quattrone, Lars Kulik, Egemen

Tanin, Kotagiri Ramamohanarao, Tao Gu - ICICS '15 Proceedings of the 10th International Conference on Information, Communications and Signal Processing.

In addition, I contributed to the following additional publications as a co-author while completing my PhD:

- **Protecting Privacy for Group Nearest Neighbor Queries with Crowdsourced Data and Computing**, Tanzima Hashem, Mohammed Eunus Ali, Lars Kulik, Egemen Tanin, Anthony Quattrone - UbiComp '13 Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing.
- **On the Effectiveness of Removing Location Information from Trajectory Data for Preserving Location Privacy**, Amina Hossain, Anthony Quattrone, Egemen Tanin, Lars Kulik - IWCTS '16 Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science.

*Dedicated to my family*

# Acknowledgements

Completing this dissertation has been a tremendous undertaking that has helped me develop both intellectually and personally. Firstly, I would like to thank my supervisors Prof. Lars Kulik and Assoc. Prof Egemen Tanin who made this research possible and provided tremendous support and guidance throughout this journey.

My sincerest gratitude and thanks goes out to my principal supervisor Lars for encouraging and convincing me to pursue graduate research when I was at a major crossroads. The continued support and advice provided over a number of years has been invaluable. I feel I have greatly benefited from Lars's mentorship in many areas. Particularly, in how to approach, define and find elegant solutions to complex problems in which I was always continuously amazed by his ability. I am thankful to Lars for all the help and support provided that has helped me develop as a researcher.

A special thanks to my co-supervisor Egemen Tanin for his strong encouragement and continued support. Particularly in ensuring this research stayed on track and progressed smoothly. I am grateful to Egemen for his continued advice and feedback that greatly helped me consistently ensure the work remained at a high standard. Egemen helped me in pushing the envelope further, ensuring experimental results hold up to scrutiny and greatly assisting me in finding solutions to complex implementation issues. I am thankful to Egemen for all the time and support offered.

I am thankful to my committee chair member Prof. Rajkumar Buyya who provided valuable advice and encouragement at key milestone reviews. I much appreciate the perspective Raj offered in how to ensure publications stay relevant and the great feedback he provided at my seminars including at my confirmation and completion seminars.

I would also like to thank Prof. Ramamohanarao Kotagiri for showing great interest and being really supportive of this research. Rao's passion and dedication to the field is really inspiring. I much appreciate the opportunity I had to work with Rao on areas of this research and value his feedback and contributions.

I am thankful to all my colleagues in the department of whom I have had the pleasure to work with both as a research student and in a more professional capacity in various teaching roles. It is great to be a part of such a talented team that is leading the way in many areas of research and to personally witness the high caliber of work that is produced frequently. A special mention and thank you to Hai Ruo Xie for providing timely and detailed feedback on key submissions.

I am very grateful and would like to thank the Department of Computing and Information Systems fantastically led by Prof. Justin Zobel and his team. I would also like to thank Ms. Rhonda Smithies and Ms. Julie Ireland for always taking the time to help me navigate many complex administrative procedures. My entire experience in higher education has been within this department and I always felt that it provided a welcoming and enjoyable learning and working environment. I am thankful to the Melbourne School of Engineering and the University of Melbourne for recognizing and supporting the great work performed.

This research was supported by the Australian Research Council and I am thankful for the full funding support. I would also like to thank and acknowledge the Defence Sciences Institute for funding a key area of my research. The funding support offered made it possible for me to pursue this dissertation and greatly appreciate the opportunity they provided me.

Finally, I would like to thank my parents Anne and Paul Quattrone and my brother Steven Quattrone for their faith and support in me throughout this journey. I will forever be grateful and indebted to my family for encouraging me and giving me the confidence to pursue academic endeavors throughout my life.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Smartphones are used frequently in everyday life and provide great convenience for users. The capabilities of smartphones have extended far beyond communications with users performing activities ranging from work related tasks including sending emails and writing documents to entertainment such as playing games and watching movies. Smartphone operating systems provide a powerful alternative to performing functions traditionally performed on a PC. Although smartphones provide great benefits, researchers have expressed privacy concerns due to the storage of sensitive data stored, such as location, on smartphones [1].

The smartphone is likely the most personal device users own and likely contains more sensitive information than any other device that users possess. This has led to privacy concerns in the event an adversary gains access to an individual's smartphone data. Smartphone related privacy threats are also becoming a conscience concern amongst the general public. On an individual's smartphone, it is common to find personal emails, banking information, browsing history, meeting schedules, location tracks, call logs, personal text messages and numerous other types of sensitive information. Furthermore, a smartphone is likely to provide a complete digital profile of a user because it synchronizes with other devices such as desktop PCs and cloud services. The scale of personal information captured daily has long been a concern of pervasive computing research community [2].

The success of smartphones combined with advancements in mobile network infrastructure has created a strong market for third party apps, cloud storage providers and location-based services (LBS) [3]. These apps and services are used frequently by smartphone users. For example, a user looking up a map would reveal their location to an LBS and a third party app used to send text messages would have access to user contacts. Mobile apps and services greatly enhance the functionality provided by the smartphone but have the consequence of causing data to be sent to these third party providers.

Mobile data privacy has become a well-researched topic in pervasive computing [4]. However, more research is needed to determine the possibility of learning personal information about a user from mobile data. For example, it may be possible to determine user encounters with other users from location data or determine the identity of a person by simply looking at the mobile sensors. We demonstrate techniques and algorithms that reveal sensitive information about a user and personal interactions based on data stored on a mobile device. It is often speculated that an analyst can infer personal information from mobile data, although literature to highlight tools and techniques utilized to invade user privacy is scarce. We aim to use raw data from a variety of sensors on a mobile device to determine user behavior.

## 1.1 Smartphones as a Data Sensor

Modern smartphones have advanced hardware capabilities embedded directly into the device that apps can leverage to provide functionality. The combination of multiple hardware features allows apps to provide advanced functionality for the user. For example, song detection apps use the microphone to detect audible songs and provide the user with the name and artist of the song. Sophisticated sensors including the accelerometer, gyroscope and temperature instruments allow a mobile phone to sense the surrounding environment. The radio capabilities allow for connecting to high-speed cellular networks, WI-FI networks and Bluetooth devices. The high quality front facing and back facing cameras allow a device to capture images and video of the environment. Location sensing with accuracies of up to five meters is possible through the combined use of inbuilt GPS and localization schemes providing known cell tower locations.

Smartphone software provides sophisticated functionality for the user via a highly accessible user interface. It has been found that users find it easier to perform certain tasks such as online browsing on smartphones than on a traditional PC [5]. The software capabilities of smartphones is now competitive to traditional PCs. For example, there is a host of office suite apps available for smartphones and tablets that provide similar functionality

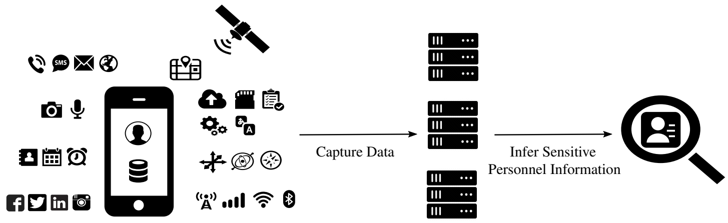



Figure 1.1: Smartphone Data Sent to Remote Servers to be Analyzed to Infer Sensitive Information

to Microsoft Office commonly used in the workplace. Currently the Android and iOS mobile and tablet operating systems are dominating the market [6]. Mobile operating systems provide a development kit and third party APIs that allow programmers to create their own applications and make use of the advanced capabilities of the device. This has led to the creation of millions of apps across a diverse range of categories including communications, games, navigation, social networks, dating, fitness and office apps.

While smartphone platforms are sold with base functionality, much of the additional functionality is provided by third-party apps. It is estimated that users spend as much as 85% of smartphone usage time interacting with apps [7]. These interactions can lead to apps storing and retaining personal information. Current platforms have made it accessible to make use of device functionality and access the data stored on a device. Early smartphone operating systems such as Symbian took a very strict approach to granting third party apps access to core functionality. For example, each time a third-party app required a location update, the user would be promoted to allow the app or disallow the app to access the GPS. The latest platforms including Android and iOS relaxed these requirements to make the platforms more developer friendly to access key data and user-friendly to not be constantly warned that certain personal information is going to be accessed. This architectural decision was a contributing factor as to why these platforms experienced great market success, however came at the expense of sacrificing user privacy.

Modern smartphones have the capability to access high-speed cellular networks and WI-FI networks that allows apps to easily download data. For example, a music app can stream directly from a cloud server. This same capability can be used to send data captured on the device itself to remote cloud servers at high speeds. Dropbox, for example provides the ability for users to backup a smartphone's photo album on to cloud storage. While this is a legitimate application, apps that do not require access to photos could easily perform the same function in the background. It is difficult for the user to detect when data transfer is occurring as no indication is given as to what data is being sent.

Software capabilities being made available to third-party developers has led to the creation of millions of apps enjoyed by users. This in turn has the side effect of the capturing and storing of user data. Captured sensitive and data considered non-sensitive can be sent to remote servers and be mined to infer personal information. In the next section, we describe how mobile data can be captured.

## 1.2 Capturing Smartphone Data

Modern smartphone platforms provide developers access to data stored locally on a device. The smartphone OS provides function or method calls that allow apps to access functionality on the device or sensitive information stored. These API calls are well documented and easily accessible by developers. Furthermore, there is also a range of undocumented calls that could be used to obtain additional information. Undocumented API calls are not intended to be used by third party apps and are often found through a process of reverse engineering. Although apps that make use of undocumented calls tend to be removed from repositories, not all apps are detected.

Third-party apps can also provide hooks for other apps to make use of the functionality they provide. This is common with social networking apps where they provide calls that can access user data stored on remote services not local to the device. This provides an additional mechanism for apps to access sensitive data. Thus, even data that is not local to the device can be accessed provided a user has social networking apps installed. An additional privacy issue with apps providing their own internal APIs is that it circumvents the smartphone's privacy protection mechanisms and relies on the developer to provide one which, may not have the same privacy guarantees.

Early smartphone platforms such as Symbian required the developer obtain a special certificate to access sensitive calls. This requirement has been relaxed by modern platforms to make it easier for developers to create apps. There are a large number of API calls that allow for easy collection of user information via public APIs. We demonstrate how to send data to a remote server without the user being aware. Furthermore, we detail a process for storing the data in a relational manner making it easy for an analyst to derive insights through methods such as joining certain data points with additional data sources. Appendix 1 describes the details of how to capture smartphone data and demonstrates how to build a data warehouse of the user data.

The ability to easily collect mobile data via third party apps can put a smartphone user's privacy in jeopardy. When combining many different types of data together and then further applying inference techniques, a more complete picture starts to emerge about an individual. Many users will not be comfortable with these profiles being created about them. Furthermore, derived profiles can affect a user's ability to access financial or insurance services given they are shared among parties if they can be accessed by the service providers.

## 1.3 Smartphone Privacy Protection

The main challenge users are experiencing in regard to securing their mobile privacy is modern platforms do not provide sufficient capabilities to safeguard privacy. Data leakage can occur through the use of third-party apps. The most popular smartphone operating system, Android, relies on a permissions system that warns users before installing an app of what types of data the app can access. However, studies show that many users do not comprehend or simply ignore warning dialogs at installation time. Dialogs have also shown to be ignored in other settings such as accepting a EULA agreement [8]. Other systems including iOS rely on app review process to determine if the types of data the app is accessing is appropriate, however the review process conducted by Apple is manual and is not made fully transparent to iOS users, thus users have to place their faith on the review process.

Once an app is installed and being frequently used by a smartphone user, there is no indication of when certain data is being accessed and if it is being sent to remote servers. Modern smartphones are secure but do not guarantee user privacy due to issues of data leakage. It is possible to access data through third party apps such as games that are popular being developed with an intention of mining for user data. Thus, app stores or repositories are commonly being used as a launching pad to collect data. Even the use of services such as a LBS or fitness tracking apps can reveal sensitive habits.

Early methods of privacy protection were restrictive, blocking any call to features that may pose a risk. For example, the device would prompt the user for permission to grant an app access to capture a location sample at each update. Modern platforms relaxed these restrictions to lower the barrier of entry for developers to create innovative apps. Features often require access to sensitive data, for example communications apps require access to the contacts. Protection apps that allow the user to block certain permissions from all installed apps have been made available. A notable example includes Stowaway [9]. In practice, this caused many apps to crash as developers do not usually guard against not being able to access sensitive calls once the user grants permission at installation time. The relaxing of these requirements has exacerbated issues pertaining to data leakage.

It is extremely difficult to access mobile data stored internally when a device is locked as the data will be wiped if the lock screen is bypassed [10, 11]. Smartphones usually make a user wait a number of minutes if too many attempts are made at entering the passcode without success. Thus, successfully attacking a passcode via brute force is not viable. In a recent case, the FBI attempted to gain access to a suspect's device to help solve an investigation. Apple refused to engineer an exploit to their device and the case received a lot of media coverage and ignited debate on whether or not Apple should unlock the phone [12], eventually a firm in Israel had an exploit that unlocked the device. Thus, it is difficult to invade user privacy via bypassing device security.

A major hurdle for smartphone users in safeguarding their privacy is there are not many configurable privacy options. Steps users can typically take to help secure privacy are to use complicated passcodes, encrypt any device storage, install a VPN and install anti-virus apps. Unfortunately, these methods do not protect the user from third-party apps that need to access private data required to provide functionality. Once the data is accessed, it can be sent to remote servers. A user may not be made aware when private information is sent to a remote service.

## 1.4 Famous Privacy Breaches

Privacy researchers have long established the risks of data being shared with third parties or made public. The problem of publishing data in a privacy-preserving manner has been found to be a hard problem to solve since enforcing really strong privacy controls can reduce the utility of the dataset to the point where it is not usable for analysis [13]. While solutions exist, there is no general solution that works well in all circumstances. Even though privacy risks exist, the benefits gained by sharing data is considered in many cases to outweigh the risks. However, in the event of a data breach, there are often serious reputational and possible legal repercussions. There are a few famous cases that have received considerable mainstream media attention involving large multinational companies.

Carrier IQ was a company that provided diagnostic information of smartphone usage to mobile service providers. The software provided was found to be capable of capturing sensitive information including text messages, retrieving call logs, recording phone calls, logging keystrokes and overriding user location settings to force capture location samples [14]. Many users were unaware that this software came preloaded with their smartphone and in many instances, it was not a straightforward process to disable or remove the software. The discovery of this software created a lot of controversy that in part eventually led to the company no longer being active.

Shanghai Adups Technology Company was recently found to have preinstalled software designed for Android smartphones. The software collects a wide array of sensitive user information similar to Carrier IQ and sends it to a remote server every 72 hours [15]. In this case, service providers were not aware of the software and it is unclear how many devices have been affected.

AOL intentionally released search data in 2006 publicly for research purposes for a large percentage of their user base. While the release was intended for academics, it was accessible to everyone. Only the user ID and the search terms were made available, search queries are sensitive and user IDs were consistent for each user. Thus, once a user ID was de-anonymized, all their search history was viewable. The New York Times successfully identified a user and even found a photo of them online [16]. Another user referred to as "*User 927*" had an unusual search history which was the subject of an online analysis [17]. The search history was even made into a play and the user was eventually identified. There is now debate as to whether or not it is ethical to use the data given it has been mirrored by many websites.

Netflix crowdsourced their recommendation system and offered a million dollar prize to whom created the best model. To facilitate this, a training and a testing dataset were released to participants. Netflix took a simple approach to securing user privacy by replacing customer IDs with random IDs. Researchers from the University of Texas were able to perform a background knowledge attack by using data from alternative movie databases including IMDB [18]. Users were de-anonymized which compromised their privacy. The privacy model employed did not take into account publicly available datasets that provide auxiliary information to perform linkage attacks that have shown to be theoretically possible in the literature.

With the advent of big data and recent commercial interest into data mining, we will likely see more privacy breaches in the future as data is more likely to be freely exchanged amongst parties or made publicly available online. There are recently available crowdsourcing services such as Kaggle [19] that allows organizations to make datasets available online to individuals from various disciplines that compete to produce the best performing model with top performers receiving a prize.

In Chapter 2, we provide details on how these breaches can be performed and possible protection mechanisms that can be employed to make it more difficult for an adversary to derive sensitive information. The interest into crowdsourcing data mining is driving the need to provide a privacy aware way of exchanging datasets. In order to make datasets privacy aware, a knowledge of what can be inferred is required which is a primary focus of this thesis. This directly affects smartphone users as it will be likely that information sourced from a device will be considered valuable for research purposes.

## 1.5 Sensitive Inferences Made from Smartphone Data

In this thesis, we present a series of algorithms and techniques to derive sensitive data from data accessible to smartphones. Our focus is on data that is considered seemingly innocuous and may be distributed freely amongst parties. We consider data to be seemingly innocuous when upon manual inspection, it appears it does not reveal much information but can reveal sensitive insights when applying additional analysis. Our approaches have advantages over existing protection models such as differential privacy [20], which only tend to work well for static datasets and are difficult to apply to dynamic datasets such as a user's mobile data that is changing constantly through continued interactions with the smartphone.

Mobile apps capture diagnostic data for the purposes of debugging and improving the quality of an app or service. In our work, we found that users can be uniquely classified using diagnostic data in isolation. Diagnostic data is considered to be less sensitive and is freely shared amongst developers. The diagnostic data of open source apps may even be available for anyone to view online. Types of diagnostic data includes the hardware information of the device, user settings and the amount of storage space remaining. Mobile apps and services often send diagnostic data to a server at regular intervals. This allows for a profile of how the users' device changes over time to be captured. We demonstrate that ensembling a number of diagnostic data points together can lead to the identification of identifying a user. We identify users with accuracies as high as 90% in a dataset containing diagnostic data. By showing that users can be identified, app developers should carefully consider the types of diagnostic data they capture.

The service providers of apps commonly store location query data inputted by the user on remote servers. An example of a location query could be *"what are the closest Italian restaurants?"*. Legislation may not permit a service provider to provide raw personal data to third parties, but may permit the service provider to exchange the query results with third parties. We were able to establish an individual's trajectory to a high degree of precision given that the individual sends continuous queries requesting the closest point of interest (e.g. restaurant) from a location service provider. The algorithm to infer the user's trajectory is based on the query result locations instead of the actual user locations.

We show that a user's personal information can be obtained via an indirect approach, i.e., the storage and exchange of query results. Experimental evaluation indicates in some cases that a user's trajectory can be determined to a high level of accuracy given a small set of points in real scenarios. We found that by simply analyzing the POI locations in the results of continuous location queries, we were able to recreate the route of the smartphone user.

The features provided by the Bluetooth wireless communications stack that come with smartphones as standard can also be used as a sensor to infer sensitive information. We ran an experiment to determine what information is inferable based on one audible Bluetooth beacon. Using our data collection app, we placed a smartphone in one location and captured RSSI data for two weeks. We were able to detect patterns of behavior of individuals when they walk past a certain room. Interestingly, we were also able to see the common times they worked during the day since some desktops seem to turn on the Bluetooth transmitter when the machine is no longer in sleep mode.

We also were able to locate indoor users and learn about their activities by capturing signal strengths of their smartphones. Bluetooth is a mature technology that facilitates communications between devices such as headsets, speakers, wearables and other smartphones.

Bluetooth capable devices in discoverable mode allow for other devices to scan and pair with it. Paired devices automatically connect to one another when they are within range. A majority of Bluetooth devices can communicate within a range of 10m. Bluetooth devices can sense the signal strength of surrounding devices by determining the Relative Strength Signal Indicator (RSSI).

By simply analyzing the signals emitted with smartphones with no other information, we found it was possible to infer location of users and possible times of when they will be at certain locations. Individuals carry their smartphone on their person in most circumstances. When smartphone users are near each other, the Bluetooth RSSI signal strength information of devices in range is available. The RSSI values between devices allows for locating users. There has been much research in proposing indoor localization schemes using Bluetooth. Outdoor localization is considered a solved problem with the use of GPS. The use of GPS indoors does not work well promoting research into indoor positioning. Localization schemes are either range-based or range-free. We propose an approach that combines both range-based and range-free methods and demonstrate how it can locate indoor users with an accuracy better than 1m. A smartphone user may not want their location inside a building such as an office complex to be exposed to others but our research shows that their smartphones can expose the information indirectly.

Smartphone users revealing their location information can also indirectly reveal potential social encounters with other individuals. Location information can be used to reveal interactions between individuals by detecting people within close proximity. An individual may not intend to reveal their personnel encounters for a variety of reasons, an example being an individual who is an employee of an organization who wishes to seek other possible opportunities may not want to let their current employer know.

We developed techniques that use location data to detect encounters between smartphone users. Encounters occur between people when they are within proximity to one another. Modern smartphones send location data continuously to services. This provides a snapshot of the location of users in near real-time. We found that this high level of granularity makes it possible to detect encounters between individuals as they occur.

Mining for encounters at scale is challenging because a distance calculation between each pair combination of individuals is required to determine the proximity between users. Nearest neighbor algorithms can be applied to this problem, however they are not efficient.

We propose the use of a constraint nearest neighbor query (c-NN) which only considers neighbors that are close. We developed an algorithm that exploits spatial properties to calculate who is within proximity efficiently. The first property we propose considers that people within proximity to one another must be within the same grid cell or surrounding grid cells. Grid cells are set to a size where people are considered to be within proximity. The second property we consider is there is a minimum time people need to be within proximity for an encounter to occur. We found that we were able to mine for the encounter patterns for millions of people in real-time.

Overall we demonstrated examples of data, which appears to not reveal much information about a user, could in fact reveal sensitive insights. We considered diagnostic data, location query results without the actual query, radio signal strength information and location traces. With these insights in mind, we aimed to develop a system called PrivacyPalisade to protect users from these insights being drawn by data miners. The system detects apps that request access permissions to data metrics that are not common for a certain type of app and alerts users before sensitive data is accessed.

## 1.6 Proposed Smartphone Privacy Protection

A secondary aim of this thesis is to provide techniques to protect smartphone users from the aforementioned inference attacks we described. At present, there are few options available to privacy conscience users that want to realize the full benefit of using smartphones while preserving their privacy. A privacy conscience user may choose not to use certain apps or perform certain functions such as using a map for navigation, however they also lose the benefit of having access to the map. We aim to balance privacy requirements with user functionality.

A key problem is measuring that if accessing certain data points is justified to provide certain functionality to the user. We developed an algorithm to detect apps that are suspiciously requesting excessive permissions. The premise is apps with similar functionalities normally share similar permissions from the user. For example, weather apps normally require access to location services but do not require access to the microphone. Our algorithm can identify outliers across each app category when viewing the permissions required by each respective app. We propose the use of Isolation Forests [21] that have demonstrated high performance in detecting outliers even when only limited training data is available. Our detection algorithm trains an Isolation Forest for each respective category. If an app in a category is requesting irregular permissions then the Isolation Forest will detect it as an outlier.

To protect user privacy, we developed a system called PrivacyPalisade. It aims to infer which apps are simply requesting certain permissions for data mining purposes and what apps actually need the data to provide functionality. The system protects users from the demonstrated privacy inferences that take seemingly innocent data and derive sensitive information. PrivacyPalisade balances privacy requirements with the functionality requirements of apps. PrivacyPalisade embeds directly into the operating system and assists the user to make more privacy focused decisions. It attempts to balance the functionality requirements of an app with the types of data an app can access.

PrivacyPalisade classifies apps as either safe, neutral or invasive. An app is safe if it does not require any sensitive permissions, neutral if the Isolation Forest does not detect it as an outlier and invasive if the app is detected as an outlier. The app launcher icons are color-coded based on the level of invasiveness determined. PrivacyPalisade displays a dialog to warn the user in the event a user opens an invasive app. Data leakage occurring from suspicious apps can also be detected by keeping a log of data points that are captured by each installed app. A smartphone user can make more privacy conscience decisions with Privay-Palisade. The user can choose whether they want to grant access to data for certain apps. We found many apps were requesting excessive permissions and conducted case studies of certain apps. Our system also suggests that smartphone users can also consider alternative apps that provide the same functionality but require less excessive permissions.

## 1.7 Research Significance

Smartphones are continuing to advance with more processing power, more sensors, faster network connections and more advanced software. There are now also additional devices that can be connected to smartphones that provide even further functionality, e.g. FM radio receiver. It is not unrealistic to consider that smartphones in the future will become the primary computing device for many users. Foreseeable development in this area increases the possibility that even more data needs to be collected by smartphones. Privacy needs to be built into device as a part of the design and not considered an add-on in order to provide effective privacy protection. Achieving this requires a strong understanding into the dangers of certain data types and what they can reveal.

Demand for data mining smartphone data is growing as smartphones reveal many insights that are useful for research. For example, the use of smartphone data to learn insights is already being performed by Apple. Users are assured their privacy will be maintained via using differential privacy mechanisms, which we explain in detail in Chapter 2. More research is needed to understand the privacy implications of exchanging mobile data and how to protect users. In this thesis, we demonstrate how data considered to be non-sensitive can infer sensitive information. The insights we provide can also help data providers carefully consider how to exchange data in a safe anonymous manner.

Smartphones that protect user privacy will be in high demand as the public becomes aware of the potential privacy implications. The early Internet faced a major barrier of users trusting it enough to perform tasks such as online shopping and banking. Eventually this was overcome with advancements in web security. Smartphones may face a similar problem in the future. A major reason as to why users appear to not be concerned about privacy is they do not comprehend the implications or are willing to sacrifice privacy for convenience.

Different users will naturally have different privacy preferences depending on a number of factors. Personality, occupation, lifestyle factors and a host of other reasons could drive user privacy decisions. For example, some users may feel comfortable with releasing user specific information such as location tracks while this may be of great concern for other users. The reasons as to what level of privacy a user expects and the types of data they wish to secure will depend on a wide range of factors. As smartphones are ubiquitous, there is a large cross section of many different types of users. Smartphone developers need to aim to satisfy a large customer base so an understanding of how to satisfy a variety of privacy preferences is needed. It would be ideal if smartphones allow users to specify what inferences they do not want to be made and then the device will ensure that relevant data cannot be leaked. Part of solving this problem is to be able to understand the inferences that can be made which this thesis contributes to this understanding.

PrivacyPalisade tackles the problem of protecting user privacy as a prototype system. Our proposals to further secure privacy can easily be adopted by smartphone manufacturers to help safeguard user privacy. Without access to research into preserving smartphone privacy, manufacturers may struggle to provide the privacy requirements a subset of their users will demand. We believe our proposed method contributes to the argument that privacy should be built in by design into the system itself.

Smartphones have become an important device for users and many users have come to heavily rely on them. While the functionality smartphones provide are important, so too is user privacy. Many privacy advocates consider the privacy concerns associated to be severe and this has put pressure on legislators to propose laws to help protect users. There is already proposed legislation that will come into effect in the EU in 2018 that will directly impact on how smartphones need to interact with users. Thus, it is important that these privacy issues are addressed to allow the smartphone technology to continue to proliferate.

## 1.8 Contributions

This thesis makes the following contributions:

1. We provide a comprehensive literature review of the current state of privacy research including dataset privacy, location privacy, trajectory privacy and smartphone privacy research. There exists a gap in the literature between more general privacy research and how it relates to smartphones. We discuss the latest privacy research and put it into context of how it relates to the use of smartphones.

2. We demonstrate an inference attack on trajectory privacy by reconstructing a route a user potentially traveled along by using query results without knowledge of the GPS tracks. The properties of a Voronoi diagram with the knowledge of when timestamped queries are used to incrementally expand out candidate paths. Weighting functions are proposed to select a potential candidate path. An accuracy level of up to 80% is achieved using the Microsoft Geolife dataset.

3. We show how Bluetooth can be used to locate users indoors with accuracies of up to 1m by combining the use of range-based and range-free localization methods. Traditionally, range-based methods use audible beacons and methods such as triangulation while range-free methods use the information provided by neighboring nodes. We demonstrate an algorithm that combines the benefits of both to yield more accurate

results. It is also demonstrated that signal data captured and stored by a Bluetooth beacon in a static location reveals sensitive information about an individual.

4. We analyzed diagnostic data commonly sent by smartphones and used it to identify users in a dataset with accuracies of up to 97%. Diagnostic data is often considered to be anonymous and even public data. While some diagnostic metrics like *manufacturer* remain unchanged, dynamic metrics such as the amount of space remaining on flash storage and user settings change overtime. We trained a naive bayes classifier and we were able to reliably identify a user with just three consecutive days of training data. This leads to the assumption of freely exchanging diagnostic data as not invasive to privacy to be challenged.

5. We developed an algorithm to infer potential encounters of an individual using a smartphone and continuous location updates to a LBS. The algorithm makes use of spatial properties that are unique to encounters. We formalize the encounter problem and propose the use of a constraint nearest neighbor query (c-NN). Unlike traditional Nearest Neighbor (NN) search that can be too slow for practical user, our algorithm is capable of mining encounters in real time.

6. We present PrivacyPalisade, a system developed for Android with the aim of protecting against privacy attacks. PrivacyPalisade integrates directly into Android at the OS level allowing it to monitor user activities and present or warn a user about certain actions. We developed an algorithm that considers app similarity when measuring for whether an app is justified to access certain device functionality. The algorithm makes use of Isolation Forests. PrivacyPalisade uses the algorithm to determine when to warn users when accessing certain apps.

## 1.9 Outline

The rest of the thesis is organized as follows:

**Chapter 2**. We provide a literature review of the current state of mobile privacy research. We also present a general background on data mining techniques and how smartphones currently provide privacy protection.

**Chapter 3**. We present an algorithm that can infer the route of users based on the query result data sent to an LBS.

**Chapter 4**. We analyze Bluetooth signals emitted by smartphones and present a localization scheme that can locate users indoors.

**Chapter 5**. We show how access to continuous location updates can be used to determine encounters patterns of individuals. We present an algorithm to mine for encounters in real-time.

**Chapter 6**. We demonstrate how smartphone users can be identified from diagnostic data.

**Chapter 7**. We present techniques on how to identify apps that are similar to one another. We also present PrivacyPalisade, a system designed to protect user privacy.

**Chapter 8**. We conclude the thesis and present the direction of future work.

**Appendix A**. We demonstrate how a smartphone developer can construct a data warehouse of user mobile data.

# Chapter 2

# Literature Review

Smartphones have proliferated in recent years providing considerable benefit to its users. The benefits have often come at the expense of user privacy either by the user directly or indirectly revealing information about themselves via interacting with the device. Research into smartphone privacy is a broad area that overlaps with other research disciplines including dataset privacy, location privacy, trajectory privacy, wireless networks and issues related to the design decisions of smartphones.

Privacy researchers are concerned with technology being used by an adversary to invade user privacy. The exposure of personal information that leads to privacy concerns fall under various types of categories including communications, financial, medical, location, political and educational. These information types can be captured and stored on a smartphone. Techniques to protect privacy draw ideas from research areas including the security, legal and database fields.

The Right to Privacy was published in 1890 and was inspired by issues of general coverage of individuals' personal lives in newspapers [22]. Laws at the time did not protect people from published privacy inferences made by the press. Journalists and photographers were free to take photos or use any other modern recording devices. The article is considered to be the foundation of modern privacy laws.

Privacy advocates argue that regulations are still not sufficient to protect user privacy and current laws are outdated relative to current technology [23]. Researchers recommend privacy be engineered directly into technology by design to protect users [24]. The foundations of information privacy research dates back to the advent of the widespread use of modern computers that enable the processing of large quantities of information and completing complex queries in mere seconds. It is now accepted that users reveal information directly by storing it on a computer and indirectly through performing tasks such as checking email or browsing the web. Smartphones are capable of performing almost all traditional computing tasks and specific tasks to mobiles such as phone calls. Potential privacy threats have been amplified by the use of smartphones.

While privacy is widely considered to be a right, other scholars argue that it is a commodity that can be transacted. Online privacy protection, for example, is currently being treated as a commodity with many online services adopting the freemium model. This is mainly as a result of relaxed regulations and will be subject to change in the future.

Privacy research is gaining more attention with revelations of government and corporate data mining programs of users' personal information. The problem of preserving privacy has been further exacerbated by the use of smartphones. Concerns expressed by early privacy researchers have come to fruition with claims of government mass surveillance programs [25]. Furthermore, agreements have been made between corporations to exchange data to drive revenue. Individuals have concern for their digital privacy but will act in ways counter-intuitive to this notion for the convenience of the services offered [26].

Analyzing datasets can reveal sensitive information about a user even when stored in just aggregate form [27]. It is common for analysts to exchange anonymous datasets for research purposes when an analysis can be of great benefit to society. Ensuring an adversary cannot infer sensitive information is a difficult problem to solve. Proposed solutions come from a variety of fields including cryptography, statistics and theoretical computer science.

Early research into privacy related to smartphones targeted location privacy as it was considered to be the number one concern preventing users from adopting smartphones [28]. The spatial nature of mobile devices coupled with advanced GPS capabilities allows for capturing accurate location samples and sending them to remote services in near real-time. Thus, individual smartphone users can be tracked and inferences from these location tracks allow for sensitive inferences to be derived. Location samples reveal sensitive insights about an individual, however an individual's trajectory has been found to be able to reveal even further insights prompting the recent research field of trajectory privacy.

Advancements in data mining techniques in combination with streamlined access to user data via smartphones allow for the discovery of intrusive insights. The adoption of privacy-preserving data mining could potentially lead to further insights being discovered in data as organizations will be better positioned to allow their data to be used without revealing it. We explore how to perform common data mining functions while considering user privacy.

Smartphone design itself impacts on the level of privacy it affords its users. Modern smartphone technology cryptographically protects data stored but does not consider privacy related issues. The success of smartphones has led to the development of millions of third party apps. The app ecosystem engineered for smartphones can lead to issues of data leakage. Every modern smartphone has Bluetooth capabilities to facilitate communications between devices, however Bluetooth has a history of being vulnerable to security and privacy threats. Malware has also been detected on modern smartphones with characteristics similar to what can be found on desktops.

Research into smartphone privacy is mainly aimed at technological solutions, however legislators have also explored enforcing protections legally. Proposed regulatory policies will influence how smartphones, mobile apps, mobile services and user interactions with smartphones as they come into effect. We describe how new regulatory policies will directly affect current platforms and how platforms will need to adapt in order to be compliant.

In this thesis, our primary focus is to determine if sensitive inferences are derivable from smartphone data that may appear to be anonymous but upon analysis can reveal sensitive insights. To the best of our knowledge, this is not covered in the literature. The work in this thesis will help fill this gap contributing to the area of understanding possible inferences and protection mechanisms.

The foundations of information privacy research highlighted many concerns related to the technological advancements in storage, retrieval and sharing of information. Smartphones with these capabilities present privacy challenges that are relevant to database technology and desktop PCs. In Section 2.1, we describe identified privacy threats and how they apply to the use of modern smartphones.

The manner in which organizations store data has shifted from internally managed databases to making use of cloud storage. Smartphones also make use of cloud storage in the form of backups or saving personal information. Statistical databases derived for lower-level records also are commonly managed by organizations. Section 2.2 describes the progression of data storage in relation to smartphones.

Smartphones contain a wealth of personal information making it an attractive option for researchers to access and collect certain types of required information. While there may be considerable benefit to the research conducted, the manner in how to collect and store this information has been the subject of much research. Section 2.3 describes potential attacks and protection models that are commonly used.

The spatial nature of smartphones presents unique privacy challenges due localization capabilities offered by the device. It is possible to learn the location history of users that in turn can reveal a wealth of sensitive information about an individual. Location privacy was once considered a key concern in preventing smartphones from proliferating. We provide an overview of the wealth of research conducted by the location privacy research community in Section 2.4.

Localization capabilities of smartphones has progressed from taking just location samples at various points in time to being able to provide continuous location updates. Continuous location updates presents further privacy challenges as additional information can be learned such as frequently used routes. This led to a new line of research in the area of trajectory privacy which is described in Section 2.5.

Advancements in data mining allow for sensitive and in many cases not obvious insights to be learned about individuals. Many of these techniques employ machine learning and artificial intelligence techniques. Smartphone data can be utilized by these data mining approaches to train models. In Section 2.5, we provide a brief overview of common data mining methods and how to apply them in a privacy preserving manner.

The architectural decisions made by smartphone manufacturers has a direct impact on how effective they are at securing user privacy. There are also a number of potential attack vectors an adversary can exploit to gain unauthorized access to a user's sensitive information. Malware is becoming increasingly more common in smartphone platforms while third party apps that provide legitimate functionality are also increasingly being used as a means of gathering data. We explore these issues in Section 2.7.

Regulatory approaches are also required to work in conjunction with technological solutions to effectively safeguard user privacy. Privacy is considered to be a UN human right and new laws coming into effect across many states. In Section 2.8, we explore various regulatory protections currently being considered and the impact they will have on the design of smartphones.

This chapter is organized as follows:

**Section 2.1** draws parallels with privacy concerns identified by early privacy research and how similar problems are arising with the use of smartphones.

**Section 2.2** details the progression of how organizations store data from internally maintained databases to cloud storage solutions. A discussion of statistical database systems is also presented which can be populated with smartphone data.

**Section 2.3** details privacy concerns related to the storage and exchanging of sensitive data. Smartphone apps are capable of sending data to databases on the cloud that can be used for research purposes.

**Section 2.4** explores location privacy in detail. Smartphones are capable of producing accurate positioning estimates and location privacy was once considered the biggest barrier facing smartphone adoption by the masses.

**Section 2.5** explores trajectory privacy which is a research discipline extending from location privacy that considers a collection of location points to reveal insights that single positioning points may not reveal.

**Section 2.6** details proposals on how to perform data mining while preserving privacy.

**Section 2.7** describes how apps can access sensitive information on the device and the privacy implications. We also explore privacy issues associated by smartphone malware and from Bluetooth capabilities.

**Section 2.8** presents regulatory proposals that will affect how smartphones are designed in the future in order for them to remain compliant.

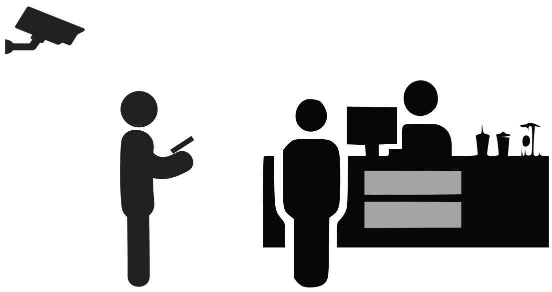


Figure 2.1: Living in a Digital World that Captures Data via Daily Interactions

## 2.1 Foundations of Information Privacy

Smartphones are the latest iteration of a long line of technological breakthroughs that had the unintended consequence of reducing privacy. Information technology is becoming more ingrained in our everyday life. We have come to not only use it ourselves but live in it through daily interactions [29]. Larry Hunter was the first to express privacy concerns from the use of desktop PCs in 1985 [30]. Nissenbaum described how information drawn from public spheres could potentially lead to privacy invasions [31].

In the 1970s, early mainframe systems with database capabilities were deployed in many organizations. Databases made it easier to store and retrieve information quickly and have replaced many paper-based methods. In the 1980s, personal computers soon followed, individuals started creating and storing their own information in digital format. The advent of the Internet in the 1990s led to the creation and sharing of information making the exchange of personal information easier than ever before.

Hunter noted that computers can join information together from multiple data sources to build an accurate profile about a person [30]. With the mainstream adoption of personal computers, users typically store sensitive information including their contacts, calendar and documents. An individual can potentially reveal personal and sensitive information that is not only captured by their computer but also systems they interact with such as mobile networks, CCTV cameras and bank cards used for transactions.

The creation of the Internet and the World Wide Web (WWW) led to the development of many web apps. Users directly or indirectly share information that is commonly captured by these services. For example, online shopping reveals purchasing behavior while instant messaging reveals personal conversations. User privacy concerns acted as a serious barrier to commercialization during the advent of the web with a study finding that users did not

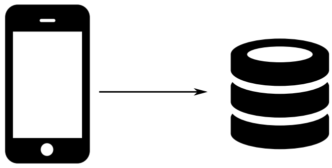



Figure 2.2: Mobile Data Stored in a Relational Dataset

enter valid information when signing up for services [32]. To make an online purchase, a user is required to reveal correct information to complete the transaction. Users were concerned with the information they need to reveal to make an online purchase.

Just as concerns were expressed with early generations of computing technology, researchers have expressed similar concerns with smartphones as they provide all the capabilities of earlier generation technology with the flexibility of being with a user at all times. A similar trend that occurred with web apps is now prevalent with smartphones as developers create mobile apps that replicate many of the web app functions. The amount of sensitive information users store digitally has increased with the advent of smartphones.

## 2.2 Datasets and Smartphones

Datasets privacy overlaps with smartphone privacy since sensitive datasets can be derived from internal smartphone data. It is likely that organizations will provide users enticing offers for access to their data. Smartphones lend themselves to be useful in capturing user data and creating a structured dataset containing a user's digital profile. Data stored internally on the device can be processed into a structured dataset for the purposes of exchange and analysis. Relational databases such as SQLite are available to install directly on a smartphone thereby simplifying the creation of internal datasets. Information rich datasets are commonly exchanged amongst organizations.

Furthermore, there are now services available that provide user backups of their smartphone data such as iCloud. While backups are convenient for a user, they also expose their personal data to the service. At present, backups are highly encrypted and cannot be accessed by the service but the user needs to trust the service does not have the decryption key.

There has been a steady progression from organizations handling their own data management to outsourcing the function to alternative providers. This has led to a great con-

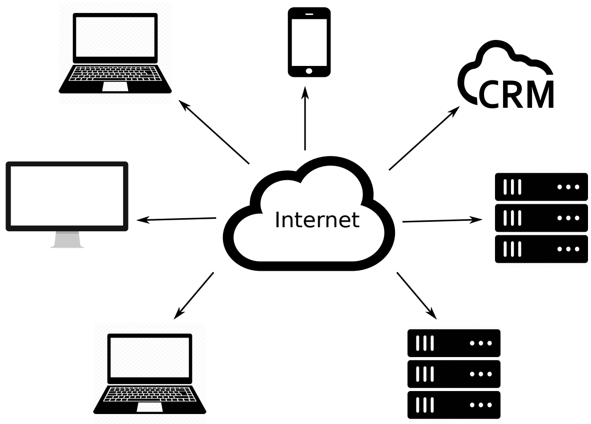



Figure 2.3: Devices Interacting with Cloud Storage Services

centration of data from a variety of sources that when combined are valuable. Datasets are made up of microdata that can be aggregated to higher orders. Statistical databases store aggregated data and are widely used to aid organizational decision-making.

In this section we discuss the progression of how organizations store data in Section 2.2.1, microdata in Section 2.2.2 and statistical databases in Section 2.2.3.

### 2.2.1 Organizational Data Storage

The manner in that organizations have handled the storage and retrieval of data has steadily changed over the last decade as a result of the development of cloud storage advancements. These advancements have led to a great concentration of data stored on cloud storage provider data centers from varied organizational sources.

Traditionally, organizational data was commonly stored locally on relational databases using servers managed by technical staff with any data the organization had collected about an individual was captured mostly via internal processes. For example, a data entry operator creating a customer entity in a CRM system. There was great cost associated in managing internal systems as hardware and software is required to be constantly updated and maintained. This led to hosting companies forming that specialize in this function.

Many companies opted to outsource their server management to benefit from operational efficiency and cost savings. This process involves a private dedicated connection being established between the host's data center and a client site. The hosting company itself had visibility of all their clients' data, although there are now ways to guard against this with encrypted file systems.

| ID | First Name | Last Name | DOB |
|---|---|---|---|
| 1 | David | Avery | 01/03/1966 |
| 2 | Claire | Scott | 15/06/1962 |
| 3 | Faith | Sharp | 07/02/1972 |
| 4 | Adrian | Reid | 23/07/1990 |
| 5 | Nathan | Miller | 31/12/1992 |

(a) Microdata

| YOB | Count |
|---|---|
| 1962 | 1 |
| 1966 | 1 |
| 1972 | 1 |
| 1990 | 1 |
| 1992 | 1 |

(b) Statistical Dataset

Figure 2.4: Aggregation of Microdata to a Statistical Dataset

Cloud services also became available to outsource hosting to remote Internet servers such as Amazon Web Services (AWS) and Microsoft Azure. This progressed further to emerging services specializing in dedicated functions such as CRM or accounting. Salesforce is a recent example of a tremendously successful online CRM system. While each client that uses Salesforce is required to enter their own data, Salesforce themselves have access to all their clients' data including from multiple direct competitors. These services have access to a rich dataset that can be exchanged or sold.

There is a legal question of who owns the data. Nevertheless, data mining is being performed on these data sets. While it is less common for detailed datasets to be exchanged, microdata and statistical datasets are frequently shared amongst organizations.

### 2.2.2 Microdata

Microdata often refers to data collected at the individual level. Some examples include age, address and occupation. Microdata being exchanged is usually protected through mechanisms of data perturbation when being exchanged. The advantage of exchanging microdata is it allows an analyst to explore the relationship between variables.

Organizations typically remove personal data on the assumption that individuals will remain anonymous. However, it is often possible to perform re-identification with linking methods [33] when access to multiple datasets is available. Attacks on datasets make use of re-linking methods that join a group of datasets together to establish a detailed profile. Proposed protection models to guard against re-linking methods have been to be effective in certain scenarios.

### 2.2.3 Statistical Databases

The widespread use of smartphones lends it to be an attractive option to collect statistical data of users for research or commercial purposes. The purpose of a statistical database is to provide aggregated information to a researcher for a set or subset of records. Consider the example in Figure 2.4, a table containing individual date of births is aggregated to provide counts of year of births. Computed statistics range from simple functions including *SUM*, *MEAN*, *COUNT*, *MIN*, *MAX* to more complicated measures such as probabilities [34].

There are major challenges in preserving privacy in statistical databases. Statistical databases are frequently used in the development of data warehouses that facilitate the generation of reports used to aid the decision-making. Data is often aggregated before being exchanged to a higher order such as a zip or postal codes with personal identifiers removed. It is commonly assumed that detailed individual data is most sensitive. The opposite can also be true as aggregated data can actually reveal further insights about an individual. Inspecting detailed data manually is laborious and it is easy to miss key details that automated processes can find. With advances in data mining technology, interactions a person makes in the public eye is far more accessible.

Turn and Shapiro described the following classifications for statistical databases: offline-online, static-dynamic, centralized-decentralized and dedicated-shared systems [35]. Statistical databases that provide realtime interaction with the user are considered to be online. In the offline scenario, the user does not know when a query is going to be processed or when it will be completed. Statistics are either static and never change or dynamic and are being constantly updated. Datasets can reside on a centralized server in one database or be split over multiple databases residing on multiple servers. Statistical database systems can be classified as a dedicated system that provides data or a shared system in which a client application will run alongside other applications. Shared systems are more prevalent in modern databases while early mainframe systems were often dedicated.

Early approaches to protect statistical database systems from users drawing sensitive inferences focused on conceptual approach, query restriction, data perturbation and output perturbation. These approaches were subject to much initial research and presented in the survey [34]. The main focus is to balance quality of output versus individual privacy for data stored in relational databases. Certain factors need to be taken into account such as the confidential requirement of certain fields and the types of queries users need to initiate. These solutions normally need to be used in conjunction with one another to provide adequate protection as not one model is sufficient for all scenarios.

The conceptual model [36] approaches the security problem by providing a framework that is implemented throughout the entire development cycle of the statistical database system. In this model, there is no query language available, limiting the user to only access statistics of certain attributes. A distinction is made between the conceptual level data and the internal data. A kernel controls access to the physical data and ensures access to confidential data is limited. This is achieved using $A$-populations that ensures at least zero or two entities are returned, preventing information about a single individual being disclosed. Noise may also be added to ensure more than one record is disclosed.

Query restriction approaches limit the type of queries users can send to a statistical database. A straight forward solution is to apply the query-set-size control method that only allows for a statistic to be released if the number of respondents that make up the aggregate satisfy a condition [37]. A query-set-overlap control can also be employed restricting the number of overlapping samples that make up successive queries [38]. While this can protect from inferences made by one user, it is not effective when multiple users collaborate to make inferences. A similar approach is to maintain an audit trail of all queries issued and look for potential compromises [39]. Populations used in statistics can be partitioned to be mutually exclusive and form what is referred to as atomic populations [40]. A user can only run statistical queries on each group in its entirety and cannot select a subset. If queries are combined then both the queries must have complete overlap. Cell suppression can be applied on static statistical databases [41]. The main idea is to suppress confidential cells that might lead to sensitive inferences being made.

Data perturbation involves transforming some of the underlying data with values that conform to an assumed probability distribution or generating statistics based on an assumed underlying probability distribution and replacing the statistical database. The most common method is to employ data swapping [42] of the original data with randomly generated data that still produces the same statistical quantities. This is only practical for static databases due to the computational requirements. Instead of swapping the data, confidential attributes can be protected by estimating a probability distribution that describes the field [43]. Care has to be taken to ensure the probability distribution does not match to closely to the original attribute otherwise exact disclosure can occur.

Output perturbation changes the output of computed statistics rather than changing the underlying data as described by data perturbation. One method is to derive a random sample set each time a query is initiated to compute the required statistics [44]. Another approach is to change the actual output of the computed statistics using an independent random variable [45]. This approach also ensures that a response will eventually be repeated. Rounding techniques can also be used to ensure that a response to a query conforms to the closest multiple of a base. Various methods that make use of rounding have been proposed [46, 47, 48, 49, 50].

Modern research in this area now tends to focus on generic dataset privacy models as opposed to securing statistical database systems. Potential dataset attacks and protection methods are presented in the next section.

## 2.3 Dataset Attacks and Protection

In this section, we provide an overview of how data is commonly stored and explore successfully performed attacks that violate user privacy. We also detail common protection methods employed including $k$-Anonymity, $l$-Diversity, $t$-Closeness and differential privacy.

### 2.3.1 Inferring Sensitive Information in Datasets

Data found on smartphones can be supplemented with publicly available databases to derive further insights that may be intrusive. Dalenius was one of the first to consider privacy in statistical databases stating that *"anything that can be learned about a respondent from a statistical database can be learned without accessing to the database"* [51]. This goal has formally been proven to be unachievable.

Even though it is well understood that exchanging statistical databases could lead to privacy invasions, there are researchers that consider the benefits of what is learned to outweigh the risk. Special care needs to be taken to ensure the probability of sensitive inferences occurring is low. There have been a number of famous cases where researchers de-anonymized datasets that were made public. Notable cases include The Netflix prize data, Massachusetts medical records, AOL search logs, Genome Wide Association studies (GWAS) datasets and mobility databases. There will likely be more cases as data mining continues to gain traction. While we detail famous cases that received public attention, there could be more many cases where inferences are being made in organizations for potential gains with general public unaware.

A major concern in all the listed attacks is that many of the datasets used were publicly available. Given an analyst performs a successful attack, if this attack becomes widely known then the users sensitive data is exposed forever due to the nature of the web. Even if data is removed online from a host, mirror sites typically cache and collect online data making it really difficult to remove from every possible source. This was especially true in the AOL and Netflix cases where search history is still available to download online from various mirrors. As more of these cases come to light, so too will the exception for organizations to take every possible step to safeguard data.

**Terry Gross's Height:** Consider the following example to demonstrate how Dalenius' goal is unachievable presented by Dwork in [20]. An adversary wishes to determine the height of Terry Gross. Assume that there exists a national database of average heights of women of different nationalities. The adversary has access to the statistical database on average heights. Auxiliary information is known that *"Terry Gross is two inches shorter than the average Lithuanian woman."*.

An adversary can learn Terry Gross's height only if he has access to both pieces of information. Having only the statistical database or the auxiliary information in isolation yields little information. Even if Terry Gross is not in the database, her height can still be determined. This is the motivating example for differential privacy, which we will explain in detail in this chapter.

**Netflix Prize:** Netflix decided to crowd source a solution for a recommendation system and offered a prize of a $1,000,000USD to the best predictive model. A training dataset was released for competing analysts. Netflix removed the data of all sensitive fields and replaced them with randomly assigned IDs.

Since Netflix is not the only online movie database, researchers joined the Netflix data to the well-established IMDB movie database to partially de-anonymize the training set [18]. This was accomplished by identifying public reviews for popular movies on IMDB and capturing all the review information. This dataset was then joined with the Netflix dataset looking for matches across the same movie and similar reviews approximately around the same time. The inference attack worked well identifying a subset of the individuals and revealing their identities. This attack was effective because people who leave reviews for movies tend to be passionate about movies and leave reviews across multiple sources.

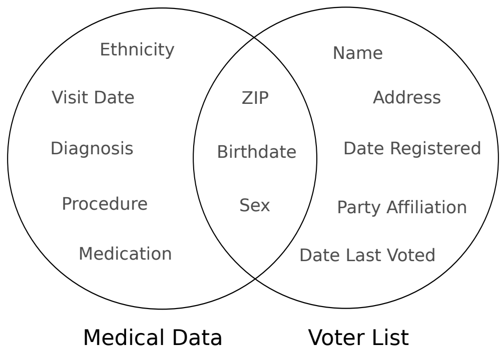



Figure 2.5: Venn Diagram Showing Linkable Fields Between Two Datasets

**Massachusetts Hospital:** A privacy attack with potentially more damaging consequences was performed on medical records. Medical data is often released under the assumption that if you remove unique identifiers such as name, address, telephone number and government IDs that patient confidentiality is maintained. Sweeney described a technique to relink data anonymized by systems including Scrub, Datafly and u-Argus [52].

In this work, the demographic data released from the 1997 Cambridge Massachusetts voting database that contained demographic data was joined to the Massachusetts hospital's anonymized database. Individuals that appeared in both databases were open to having certain medical attributes identified. The governor of Massachusetts was identified using this approach. The approach is illustrated in Figure 2.5. Attacks such as this are particularly worrying because this type of information exposed could effect a person's ability to receive health insurance or even effect career prospects.

**AOL Search:** There is numerous studies supporting the assumption that search history is personal and can reveal a user's identity and private information. Search history can easily be found on a user's smartphone by opening up their web browser. AOL Search ignored these risks and released search logs for over 650,000 users covering a three month period containing every search query. The search queries had a unique ID for each user, thus if a user was identified then their entire search history could be discovered. The release was only intended for academics but was publicly placed online with no controls.

Even though the mistake was acknowledged and removed in a few days, the dataset was mirrored on many websites. A user was identified and exposed in an article with permission by the New York Times, there have also be claims of other users being identified.

| Scenario | Datasets | Technique | Inferred |
|---|---|---|---|
| Finding Height [20] | National Heights | Auxiliary | Height |
| Netflix Prize [18] | Netflix/IMDB | Linkage | Identity |
| Hospital Records [52] | Medical Records/Voter List | Linkage | Identity |
| AOL Search [16] | AOL Search History | Linkage | Identity |
| GENOME [53, 54] | Genotype Statistics | Auxiliary | Identity |

Table 2.1: Summary of Demonstrated Dataset Inference Attacks

**GENOME Projects:** Medical research into potential links between human genomes and common diseases is prevalent and receives a significant amount of attention. Research into this area could yield important information into improving treatments so there is considerable benefits to society into finding potential associations.

The National Institute of Health (NIH) publishes allele counts based on certain genotypes of individuals that have an ailment and a control group of those that do not. It was found in [53, 54] that individuals participating in a case group can be identified as having an ailment when all participants in the case group had the ailment. Due to privacy concerns, the NIH had to discontinue public access to these datasets. DNA just like fingerprints change very little over time so once it is leaked, it can cause significant breaches in privacy.

Smartphones already collect fingerprint information and will likely collect additional biometric data such as an IRIS scan as they continue to advance.

### 2.3.2 $k$-Anonymity Protection

The principal of $k$-Anonymity states that *"the information for each person contained in the release cannot be distinguished from at least $k-1$ individuals whose information also appears in the release"* [33]. There are properties that exist in a dataset that that make it more difficult to perform linkage attacks, $k$-Anonymity being one such property. The property of $k$-Anonymity is frequently used when protecting datasets and ensuring user privacy when accessing systems such as an LBS.

$k$-Anonymity defines protections for quasi-identifier attributes. The term quasi-identifier was defined in [55] to represent potentially sensitive attributes. Attributes in a dataset are considered quasi-identifiers if they are not unique identifiers but can be combined with other attributes to identify an individual. In order to make a dataset $k$-Anonymous, quasi-identifiers need to be generalized or suppressed. Suppression involves removing quasi-identifiers, curators commonly place a '*' character in its place. Generalization methods

| Zip Code | Age | Gender | Condition |
|---|---|---|---|
| 3025 | 34 | Male | Heart Disease |
| 3020 | 39 | Male | Heart Disease |
| 3021 | 33 | Female | Heart Disease |
| 3020 | 31 | Female | Heart Disease |
| 3065 | 54 | Male | Viral Infection |
| 3065 | 46 | Male | Heart Disease |
| 3087 | 22 | Male | Heart Disease |
| 3088 | 25 | Male | Viral Infection |

(a) Medical Data

| Zip Code | Age | Gender | Condition |
|---|---|---|---|
| 302* | 3* | Male | Heart Disease |
| 302* | 3* | Male | Heart Disease |
| 302* | 3* | Female | Heart Disease |
| 302* | 3* | Female | Heart Disease |
| 306* | $\geq 40$ | Male | Heart Disease |
| 306* | $\geq 40$ | Male | Viral Infection |
| 308* | 2* | Male | Heart Disease |
| 308* | 2* | Male | Viral Infection |

(b) 2-Anonymous Table

Figure 2.6: Medical Data Anonymized

bucket the data range. For example, individuals with an age under 30 can be listed as $\leq 30$ as opposed to stating the actual age.

To prevent re-linking of data, $k$-Anonymity offers scientific guarantees. Various systems that make use of $k$-Anonymity have recently been patented. Consider a curator releasing a dataset where typical identifiers are removed such as name and government issued ID numbers. However, there exists other types of attributes such as age, sex, zip code that can be obtained from public records and be joined to an anonymized dataset to identify individuals. Sweeney proposed a formal framework to evaluate algorithms used to generate a release of information designed to anonymize data. The following definitions proposed by Sweeney:

**Definition 2.3.1.** Attributes Let B$(A_1, , A_n)$ be a table with a finite number of tuples. The finite set of attributes of B are $\{A_1, , A_n\}$.

**Definition 2.3.2.** Quasi-identifier Given a population of entities $U$, an entity-specific table $T(A_1, , A_n)$, $f_c : U \rightarrow T$ and $f_g : T \rightarrow U'$, where $U \subseteq U'$. A quasi-identifier of $T$, written $Q_T$, is a set of attributes $A_i, , A_j \subseteq A_1, , A_n$ where: $\exists p_i \in U$ such that $f_g(f_c(p_i)[Q_T]) = p_i$.

**Definition 2.3.3.** $k$-Anonymity Let $RT(A_1, ..., A_n)$ be a table and $QI_{RT}$ be the quasi-identifier associated with it. $RT$ is said to satisfy $k$-Anonymity if and only if each sequence of values in $RT[QI_{RT}]$ appears with at least $k$ occurrences in $RT[QI_{RT}]$.

With these definitions, we can ensure tables are $k$-anonymous before that are exchanged. The higher value of $k$, the greater the privacy. Consider Figure 2.6a that contains medical information for patients and the 2-anonymous version in Figure 2.6b. Suppression is applied to the zip code and age fields. Generalization is also applied to ages over 40 by simply marking them as $\geq 40$.

Algorithms have been proposed to ensure datasets are $k$-anonymous [56, 57, 58, 59, 60, 61, 62]. Further research into $k$-Anonymity found drawbacks of using the model for privacy. Models using $k$-Anonymity are susceptible to inference attacks as they attempt to retain partial information about certain dimensions. It was demonstrated in [63] that anonymity is hard to achieve using $k$-Anonymity when there are a large number of quasi-identifiers. Information loss increases steadily as dimensionality increases. The experiments showed that for Gaussian clusters and $k = 2$, anonymity was not maintained for over 20 attributes and information loss was unacceptable for data mining purposes.

It was also demonstrated in [64, 65] that solving for $k$-Anonymity even for simple cases of $k = 2$ is an NP hard problem. An algorithm was proposed using a heuristic that produced good results in a reasonable running time. In addition to issues in optimally achieving $k$-Anonymity, datasets are susceptible to linking attacks that $l$-Diversity attempted to address.

### 2.3.3 $l$-Diversity Protection

$l$-Diversity has been proposed to overcome some of the shortcomings discovered with $k$-Anonymity. $l$-Diversity proposes that $k$-Anonymity is maintained as well as ensuring there is diversity amongst the sensitive values [66]. Algorithms used in $k$-Anonymity can be conveniently adapted to enforce $l$-Diversity with little effort.

**Definition 2.3.4.** $l$-Diversity *A $q^*$-block is L-Diverse if contains at least l "well-represented" values for the sensitive attributes $S$. A table is L-Diverse if every $q^*$-block is L-Diverse*

The authors also expanded on the $l$-Diversity definition to provide variants that can be more suitable depending on the dataset. These proposals include entropy $l$-Diversity and recursive $(c, l)$-Diversity. Entropy $l$-Diversity ensures that sensitive values are distributed to satisfy $log(l)$ in each equivalence class.

**Definition 2.3.5.** Entropy $l$-Diversity *A table is Entropy L-Diverse if for every $q^*$-block*

$$-\sum_{s \in S} p_{(q^*,s)} \, log(p_{(q^*,s)}) \geq log(l)$$

*where $p(q^*) = \frac{n_{(q^*,s)}}{\sum_{s' \in S} n_{(q^*,s')}}$ is the fraction of tuples in the $q^*$-block with sensitive attribute value equal to $s$*

| Zip Code | Age | Gender | Condition |
|---|---|---|---|
| 302* | 3* | Male | Heart Disease |
| 302* | 3* | Male | Heart Disease |

(a) Homogeneity Attack

| Zip Code | Age | Gender | Condition |
|---|---|---|---|
| 308* | 2* | Male | Heart Disease |
| 308* | 2* | Male | Viral Infection |

(b) Background Attack

Figure 2.7: Attacks on $k$-Anonymity

Recursive $(c, l)$-Diversity ensures that the most frequent value can not be determined to appear frequently.

**Definition 2.3.6.** Recursive $(c, l)$-Diversity *In a given $q^*$-block, let $T_i$ denote the number of times the $i^{th}$ most frequent sensitive value appears in that $q^*$-block. Given a constant c, the $q^*$-block satisfies recursive $(c, l)$-Diversity if $r_1 < c(r_l + r_{l+1} + ... + r_m)$. A table $T^*$ satisfies recursive $(c, l)$-Diversity if every $q^*$-block satisfies recursive $l$-Diversity. We say that 1-Diversity is always satisfied.*

The use of $k$-Anonymity quickly gained popularity and many algorithms were proposed to process datasets to be $k$-Anonymous. Even though $k$-Anonymity provides better protection by reducing the granularity of quasi-identifiers, it does not protect sensitive attributes that are the focus of the dataset opening an avenue for inference attacks. The homogeneity attack and background knowledge attack were presented in [66].

The homogeneity attack can be performed when a dataset contains a group where the sensitive value is the same for $k$ records. An adversary may know each quasi-identifier of a particular individual allowing them to learn the sensitive value. Consider the example in Figure 2.7a, a health insurance company may have auxiliary knowledge of what customers are in a research dataset. They can link their customer database using a fuzzy match with quasi identifiers with health research datasets and determine with certainty that individuals that fall within a certain group have a condition.

Background knowledge of quasi-identifiers can be used by an adversary to deduce between two or more possible options in a group. The example used in [66] and shown in Figure 2.7b demonstrated a Japanese patient was identified to be in a group that listed two possible conditions being *"Heart Disease"* or *"Viral Infection"*. The knowledge that heart disease is not prevalent amongst Japanese patients allows an adversary to learn the condition is a *"Viral Infection"* with a high probability of being correct. Figure 2.8 demonstrated a table that ensures the sensitive attribute being *Disease* is $l$-diverse.

| Race | Zip | Disease |
|---|---|---|
| Caucas | 787XX | Flu |
| Caucas | 787XX | Shingles |
| Caucas | 787XX | Acne |
| Caucas | 787XX | Flu |
| Caucas | 787XX | Acne |
| Caucas | 787XX | Flu |
| Asian/AfrAm | 787XX | Flu |
| Asian/AfrAm | 787XX | Shingles |
| Asian/AfrAm | 787XX | Acne |
| Asian/AfrAm | 787XX | Flu |
| Asian/AfrAm | 787XX | Acne |
| Asian/AfrAm | 787XX | Flu |

Figure 2.8: Quasi-Identifier Equivalence Class Must Have Diverse Sensitive Attributes(s)

| Zip Code | Age | Salary | Disease |
|---|---|---|---|
| 47677 | 29 | 3K | Gastric Ulcer |
| 47602 | 22 | 4K | Gastritis |
| 47678 | 27 | 5K | Stomach Cancer |
| 47905 | 43 | 6K | Gastritis |
| 47909 | 52 | 11K | Flu |
| 47906 | 47 | 8K | Bronchitis |
| 47605 | 30 | 7K | Bronchitis |
| 47673 | 36 | 9K | Pneumonia |
| 47607 | 32 | 10K | Stomach Cancer |

(a) Original Salary/Disease Table

| Zip Code | Age | Salary | Disease |
|---|---|---|---|
| 47677 | 2* | 3K | Gastric Ulcer |
| 47602 | 2* | 4K | Gastritis |
| 47678 | 2* | 5K | Stomach Cancer |
| 47905 | $\geq 40$ | 6K | Gastritis |
| 47909 | $\geq 40$ | 11K | Flu |
| 47906 | $\geq 40$ | 8K | Bronchitis |
| 47605 | 3* | 7K | Bronchitis |
| 47673 | 3* | 9K | Pneumonia |
| 47607 | 3* | 10K | Stomach Cancer |

(b) A 3-diverse Version

Figure 2.9: Dataset Transformed to be 3-diverse

### 2.3.4 $t$-Closeness Protection

In $t$-Closeness, distributions between sensitive values are required to be within a threshold $t$. The $l$-Diversity principle was further refined using an additional principle called $t$-Closeness [67]. Researchers noted that the distribution of sensitive values also needs to be taken into account. A trade off with using $l$-Diversity is it treats all values in a sensitive field the same, which could reduce effectiveness of data mining algorithms.

**Definition 2.3.7.** $t$-Closeness *An equivalence class is said to have $t$-Closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold $t$. A table is said to have $t$-Closeness if all equivalence classes have $t$-Closeness.*

$t$ is set to the value enforced on each equivalence class using the Earth-Mover metric. The distance between distributions can be calculated via Earth-Mover metric [68], frequently used within the image-retrieval community to perform image similarity analysis.

The Earth-Mover metric takes a feature space and quantifies the minimum amount of work to transform one feature space to another. Consider the space $P$ and $Q$, the ground distances between $P$ and $Q$ is defined in the matrix $D = d_ij$. $F = [f_ij]$ is defined as the matrix that minimizes the overall cost between the flow of $P$ to $Q$.

**Definition 2.3.8.** Earth Mover's Distance

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} f_{i,j} d_{i,j},$$

$$f_{i,j} \geq 0, 1 \leq i \leq m, 1 \leq j \leq n$$

$$\sum_{j=1}^{n} f_{i,j} \leq w_{pi}, 1 \leq i \leq m$$

$$\sum_{i=1}^{m} f_{i,j} \leq w_{qj}, 1 \leq j \leq n$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{i,j} = \min \big( \sum_{i=1}^{m} w_{pi}, \sum_{j=1}^{n} w_{qj} \big)$$

$$EMD(P,Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}}$$

While $l$-Diversity overcomes limitations of $k$-Anonymity, it suffers from its own limitations being that $l$-Diversity could be hard to achieve for some datasets, may be insufficient to prevent attribute disclosure. Consider the case where a dataset only has one attribute indicating the test result for a particular virus with 99% of the samples being negative and 1% being positive. A test such as this has very different sensitivity within the attribute depending on the result. A patient may not mind disclosing a negative result but would not want to disclose a positive result. Ensuring a 2-diverse table would result in heavy information loss.

Attacks can also be performed on $l$-Diverse tables. The skewness attack and similarity attack were demonstrated in [67]. When an attribute is heavily skewed, $l$-Diversity cannot prevent attribute disclosure effectively. In an actual dataset containing the test results of a disease, only 1% suffered from the ailment. However, enforcing 2-Diversity implies that anyone in the resulting dataset has a 50% chance of testing positive. Thus, using $l$-Diversity under this scenario leads to a greater chance of a privacy breach occurring on the released dataset.

A similarity attack can be performed when the attributes are distinct but semantically

| Zip Code | Age | Salary | Disease |
|---|---|---|---|
| 4767* | $\leq 40$ | 3K | Gastric Ulcer |
| 4767* | $\leq 40$ | 5K | Stomach Cancer |
| 4767* | $\leq 40$ | 9K | Pneumonia |
| 4790* | $\geq 40$ | 6K | Gastritis |
| 4790* | $\geq 40$ | 11K | Flu |
| 4790* | $\geq 40$ | 8K | Bronchitis |
| 4760* | $\leq 40$ | 4K | Gastritis |
| 4760* | $\leq 40$ | 7K | Bromchitis |
| 4760* | $\leq 40$ | 10K | Stomach Cancer |

Figure 2.10: Table that has 0.167-Closeness w.r.t Salary and 0.278-Closeness w.r.t Disease

imply something similar, consider Figure 2.9. If an individual belongs to one class where all values of another class are the same then a sensitive value could be implied. For example, an individual may be in a dataset where the sensitive attributes are *"Salary"* and *"Disease"*. All rows where *"Salary"* is below 3k only have stomach related diseases. Thus, if you knew an individual was in the dataset and they earn below 3k, it is implied they have a stomach related disease. This type of attack is similar to the homogeneity attack described earlier.

Using $t$-Closeness provides an upper bound between the distributions of sensitive values between the real and anonymized dataset and is considered to be more effective than many other privacy preserving techniques for numeric attributes [69]. Figure 2.10 demonstrated how $t$-closeness is enforced for the dataset we discussed in Figure 2.9a.

### 2.3.5 Differential Privacy Protection

The aim of differential privacy is to maximize individual privacy in a statistical database while still allowing the dataset to provide accurate information. Apple have recently announced that they will be adopting differential privacy and building it into iOS. Differential privacy allows researchers to exchange statistical databases with high utility while maintaining the privacy of participants. The definition used in differential privacy can be applied in both interactive and non-interactive settings. This is achieved through very rigorous mathematical definitions inspired from work in the cryptography community.

A curator using differential privacy can provide a guarantee that the ability to infer any individual in the set is the same regardless if they opt in or out of the dataset. Furthermore, an answer provided by an individual in the dataset has no significant impact on the released results relative to other participants. Differential privacy aims to ensure that the insertion or removal of any element in a database has a low chance of influencing the outcome of any

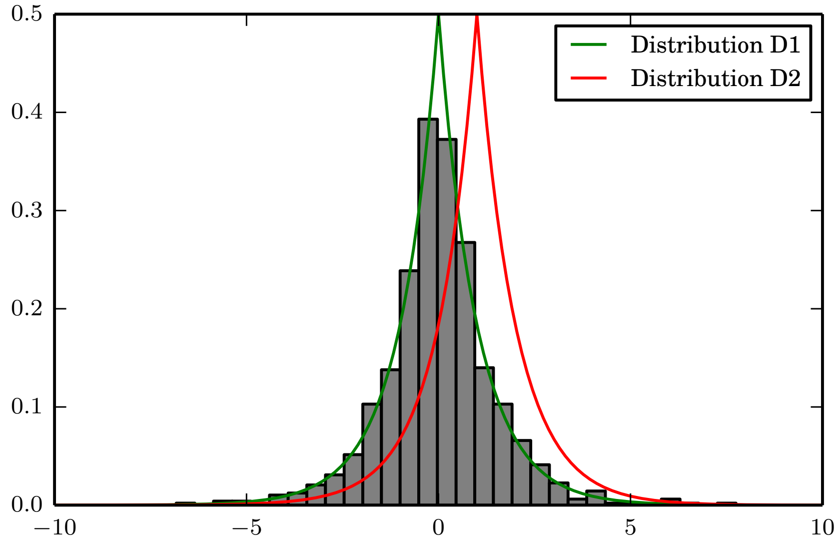



Figure 2.11: Adding Noise via the Laplace Distribution to Satisfy Differential Privacy

analysis. When joining a database onto other databases, no risk is incurred of additional information being inferred. It is assumed that information that is distributed could be used to reveal other types of information about an individual.

To ensure the probability of learning about each individual dataset is the same, we can check the output of each privacy function that exports data and see how the probability can change with the insertion or removal of individual rows. Consider the pair of databases $(D, D')$ where they differ by only one row, thus one database is the subset of the other database that is larger.

Assume databases $D_1$ and $D_2$ contain a set of rows that differ by at most one element. Consider an algorithm a curator applies to share a dataset using the randomized function $\mathcal{K}$. The function is applied to so that the data can be released with privacy protections in place. The input is the data set while the output is the released information. Differential Privacy is achieved if the following definition can be satisfied:

**Definition 2.3.9.** Differential Privacy Requirement A randomized function $\mathcal{K}$ gives $\epsilon$-differential privacy if for all data sets $D_1$ and $D_2$ differing on at most one element, and all $S \subseteq Range(\mathcal{K})$,

$$Pr[D_1 \in S] \leq exp(\epsilon) \times Pr[D_2 \in S]$$

where the probability space in each case is over the coin flips of $\mathcal{K}$.

Various algorithms $\mathcal{K}$ have been proposed to satisfy the differential privacy requirements though techniques such as adding noise [70, 71]. While Differential privacy does not provide an absolute guarantee, it does provide a very strong guarantee as the probability of information being inferred is a function of a pre-determined probability.

The Laplace mechanism [70] demonstrated in Figure 2.11 was the first method proposed by the same authors that formally defined differential privacy. The main idea is to add noise as per the Laplace distribution in areas of a dataset that is not differentially private. Approximatively $1/\epsilon$ multiplied by the effect of any individual in the set can have on an outcome leads to $e^{\epsilon} \approx (1 + e)$.

The exponential mechanism was later proposed and allows for more general properties of the dataset such as categorical values to be maintained [71]. For example, a curator may wish to a dataset of eye colors of the group of people in a way that allows queries such as what is the most common eye color without revealing the eye color of any participant. The exponential mechanism ensures that the probability of queries such as the one aforementioned being answered correctly increases exponentially.

## 2.4 Location Privacy

Smartphones are used frequently to capture location samples of a user. Thus, the study of location privacy is crucial in understanding of smartphone privacy issues. Location privacy has developed in recent years into a well-researched area with protection techniques available to keep user location data private as well as providing an understanding of the types of location attacks that are possible. Smartphones with GPS capabilities can capture location samples with accuracies as high as 5m and can also determine location from coarse location data such as cell tower and WI-FI point locations. Location tracks are a quasi-identifier [72] due to the possibility of being able to identify a user and therefore are considered to be sensitive information.

The initial motivation for location privacy research was users exposing their location was considered a barrier for location technologies such as smartphones to proliferate. Duckham and Kulik defined location privacy as *"a special type of information privacy which concerns the claim of individuals to determine for themselves when, how, and to what extra location information about them is communicated to others"* [73]. Current smartphones do not satisfy location privacy by this definition. The authors also formally defined obfuscation [74, 75] that aims to protect user location privacy by degrading the quality of location information. The idea of obfuscation refers to degrading the quality of positioning estimates. Inaccuracy was defined by the authors as providing a positioning estimate that is different from the actual estimate and imprecision was referred to as meaning multiple possible locations.

An individual revealing their location can put them in serious danger if they are being maliciously targeted. They may be exposed to unsolicited advertising or denied insurance. For example, a health insurance provider may deny cover if it is known that a user visits the doctor frequently under the assumption they will make a claim. More serious consequences include the possibility of an individual falling victim to physical abuse or having their reputation damaged from the exposition of socially compromising areas they may visit.

There has been a number of location inference attacks demonstrated that can violate an individual's location privacy, which we will explore in Section 2.4.1. Protection methods including anonymity, obfuscation and the use specialized query processing. Obfuscation is explored in Sections 2.4.2 and Section 2.4.3. Spatial cloaking is detailed in 2.4.4. Location based service architectures are covered in Section 2.4.5. Application specific queries are discussed in Section 2.4.6. Users often want to exchange utility for privacy with approaches with suggestions on how to achieve this explored in Section 2.4.7. Measuring location privacy is a difficult problem explored in Section 2.4.8. Numerous studies have also been performed on attitudes towards location privacy discussed in Section 2.4.9.

### 2.4.1 Inference Attacks using Location

Sensitive inferences about an individual can be made from analyzing their location data. There may be places a person visits that they consider private and do not want to share. For example, an employee may not want to let their employer know they are being interviewed for a job at a competitor during their lunch break. Manually analyzing individual location samples can lead to intrusive assumptions being made. We focus on algorithmic methods that analyze location data to revealing inferences that are not obvious from simple inspection. These are referred to as *inference attacks* in the literature.

Modern production LBSs offer little privacy protection despite there being a wealth of location privacy research demonstrating privacy preserving methods. Location inference attacks are developed by the community to promote the understanding that stronger protections should be adopted by LBS providers. There are multiple attack vectors an adversary can explore to draw sensitive inferences. Most inference attacks take advantage of movement patterns to draw inferences. Attacks demonstrated in the literature assume some level of protection such as pseudonymity being enforced. Pseudonymity is the most common method to preserve anonymity for location tracks [76] and requires identifiers to be replaced with an unidentifiable value.

It was demonstrated in [77] that even when a pseudonym was not present, location tracks could still be correlated to specific users. The authors applied a modified version of Reid's algorithm that is used for multi target tracking [78]. The algorithm makes assumptions and re-evaluates them when new information becomes available. The authors also successfully applied the attack on real would GPS data [79]. An alternative approach demonstrated in [80] made use of side information such as social media checkins. It was found that only 10 pieces of information were required to identity 30%-50% of an individual's path. Another attack that assumed the use of a location based social network used relative distances. The strategy used was to fake positioning estimates at different locations in order to obtain relative distances between an adversary and a user at multiple locations allowing for a triangulation attack to be performed [81].

Interestingly in [82], the authors assumed that location data would be available for a short amount of time to train a predictive model for data mining purposes. This process is typical in the data mining community. By using a Markov chain model, it was found even when the data underwent sanitation via adding spatial and temporal noise. Social media check-in data containing time and location was used in [83] to train a model that could infer detailed demographic information to a high accuracy. The authors showed how their model could successfully predict age, gender and education.

A study published in Nature analyzed the movement data for one and a half million individuals collected over fifteen months [84]. It was discovered that human mobility traces are highly unique and just four spatio-temporal points are needed to uniquely identify 95% of individuals. Even coarser datasets offer individuals little anonymity. The cell tower information and location were recorded each time a user made a phone call or sent a text message. Cell tower locations are coarse approximations while the latest location positioning technology can send constant location updates at a much higher granularity and frequency.

Significant places can be learned from GPS track data. Potential attacks on location tracks were being considered as early as the year 2000 when the technology was being adopted for commercial use. The limitation of GPS only working outdoors was exploited to determine buildings users visited in [85]. Location samples were considered significant where GPS signal was lost three or more times in a given radius. This work was extended to consider additional variables factoring in time e.g. the duration of the GPS reporting at low accuracy [86]. It has been shown that obfuscation techniques can protect home locations [87], however the authors in [88] found also knowing work location can be used to identify a user. External census data was used in to demonstrate that individuals that live and work in different census blocks are much less anonymous than those that do. Another study [89] used call logs and cell tower locations to infer that the top two to three locations calls are made can be used to uniquely identify an individual. This finding held true even when data was made more coarse than zip code regions.

A varied approach proposed the use of $k$-means clustering [90] by specifying a radii and considering the centroids of the clusters as important places. A more advanced method using time based clustering was shown in [91] considering how clusters can change over time to further refine significant places. A rigorous model formally defining stay locations and destination locations and associated algorithms was presented in [92]. Experiments demonstrated the approach was effective in extracting the locations of a user's home, work, gym and frequented shopping malls.

Location traces collected by a LBS can reveal the home address of a user. The authors in [93] captured 239 GPS tracks collected over a week from drivers from Detroit. They examined a subset of 65 drivers and were able to determine the participants' home to an accuracy of 85%. The algorithm worked by filtering out all locations samples that were high speed, select a target region of interest and drop samples outside the region, apply $k$-means clustering and use the resultant centroids as potential home candidates. A candidate was selected based on heuristics including arrival time and zoning information.

Krumm presented a similar approach to determine home locations in [87]. The study included 172 volunteers that agreed to provide two week's worth of GPS data using freely available tools. Three criteria were considered including the trip must have at least ten measured points, length of trip be at least one kilometer long and one pair of points was traveling at least at 25 miles per hour, thus removing noisy trips from the GPS tracks. A heuristic is then applied that considers the last destination, weighted median, largest cluster and best time. The identity of several volunteers was successfully identified by using an online white pages lookup that provides a reverse geocoder service.

High-level behaviors and user context can be determined by analyzing location tracks. The mode of transportation can be determined by training a model using Bayesian filtering [94]. High-level information that is inferable includes the prediction of trips [95, 96]. These studies demonstrated the viability of determining context and has led to a new line of research. Even the mental health of individuals can be determined, the authors in [97] trained a model to predict when a user is depressed to a high accuracy. Features such as the number of different places and the number of significant places were used.

Background knowledge of the maximum velocity of a given road network has been used in maximum movement bound attacks. It has been demonstrated that user locations and trajectories can be inferred. With speed bounds it was shown that a user's location must be inside the overlapping regions of the bounds [98]. Trajectories were reconstructed in [99] by exploiting known distances of the trip shared for data mining purposes to find candidate paths that satisfy the distance length. Spatial nearest neighbor queries have suggested to cloak a user's location from LBSs [100]. However, privacy concerns with spatial cloaking have been demonstrated using maximum possible bound attacks using GPS tracks [101, 102].

In a recent attack performed by our lab, GPS turn instructions were demonstrated to be useful in reconstructing a user's original trajectory [103]. The approach derived every possible route in a road network given a set of turn instructions and distances. An incremental approach was used by first traversing each connected edge of every possible source. If a match was found, then the path would continue to be expanded, otherwise pruned. Each time the traversal led to a complete match, it was stored. In the event only one path was found then the user's route was disclosed. In the event a few routes were found, it was still possible to infer the correct path with using additional background knowledge.

In Chapter 3, we present our maximum movement attack. The location of the query results are utilized without knowledge of the location of the query. We highlight the dangers of assuming data is anonymous even if the location information of the query is removed.

| Study | Dataset | Technique | Inferred Information |
|---|---|---|---|
| Marmasse, 2000 [85] | GPS tracks | Clustering | Significant Places |
| Ashbrook, 2003 [90] | GPS tracks | Predictive Modeling | Significant Places |
| Patterson, 2003 [94] | GPS tracks | Predictive Modeling | Transportation Mode |
| Marmasse, 2004 [86] | GPS tracks | Clustering | Home Location |
| Kang, 2004 [91] | GPS tracks | Clustering | Significant Places |
| Hariharan, 2004 [92] | GPS tracks | Predictive Modeling | Home Location |
| Gruteser, 2005 [77] | GPS tracks | Linear modeling | Pseudonym Correlation |
| Hoh, 2005 [79] | GPS tracks | Multi Target Tracking | Pseudonym Correlation |
| Hoh, 2006 [93] | GPS tracks | Clustering | Home Location |
| Cheng, 2006 [98] | GPS tracks | Trajectory Tracing | User Location |
| Mokbel, 2006 [101] | GPS tracks | Maximum Bounds | User Location |
| Krumm, 2007 [87] | GPS tracks | Clustering | Home Location |
| Froehlich, 2008 [95] | GPS tracks | Predictive Modeling | Trip Prediction |
| Krumm, 2008 [96] | GPS tracks | Markov Model | Trip Prediction |
| Ghinita, 2009 [102] | GPS tracks | Maximum Bounds | User Location |
| Golle, 2009 [88] | US census data | Background Knowledge | User Location |
| Kaplan, 2010 [99] | Path distances | Multilateration | User Location |
| Zang, 2011 [89] | Call meta-data | Background Knowledge | User Location |
| Ma, 2013 [80] | GPS tracks | Background Knowledge | User Location |
| Gambs, 2014 [82] | GPS tracks | Predictive Modeling | User Location |
| LiZhu, 2014 [81] | Proximity measures | Triangulation | User Location |
| Canzian, 2015 [97] | GPS tracks | Predictive Modeling | User Location |
| Zhong, 2015 [83] | Location check-ins | Tensor Model | User Location |
| Hossain, Quattrone, et al., 2016 [103] | GPS tracks | Road Network Traversal | Trip Prediction |

Table 2.2: Summary of Location Inference Attacks

### 2.4.2 Obfuscation via Pseudonyms to Achieve Anonymity

Anonymity based techniques aim to prevent linkages between location samples and user identities. Pseudonyms [76] are a value used to replace sensitive key identifiers such as a name or address. Often identifiers are omitted and replaced with asymmetric hashes to produce a unique ID. This approach is easy to implement and most commonly employed as a protection mechanism when exchanging location data.

Simply applying consistent pseudonyms does not safeguard against an adversary collecting a history of user data. It is often possible to identify a user and their patterns of behavior given enough location samples are available. For example, consider a user with a fixed pseudonym ID constantly visiting the same residential and office locations. Individuals are often unique when viewing home/work location pairs [88] implying this background knowledge can identify individuals. One approach to remedy this is to replace the pseudonym of a user often. However, pseudonym linkage is still possible when user preferences are stored by the LBS [104]. It is also possible to link pseudonyms by analyzing locations that a user occupies for long periods of time, particularly in residential locations.

Beresford and Stajano introduced *"mix zones"* [105] as a means of securing pseudonyms. Instead of assigning a pseudonym to a user identity, pseudonyms are assigned based on re-

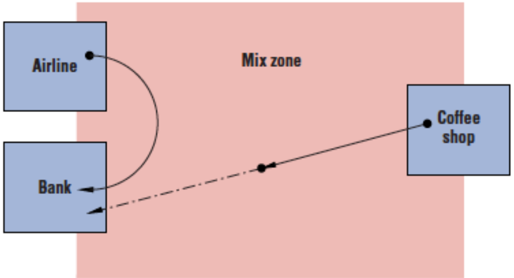



Figure 2.12: Mix Zones for Protecting Location Privacy [104]

gions such as a shopping center or an airport. This makes it difficult to identify a user because a pseudonym can be assigned to anyone within the *"mix zone"*. Consider the example in Figure 2.12, it may be possible to infer when a user travels from the coffee shop to the bank. Therefore, any pseudonyms of users within that general region are mixed to preserve privacy. The effectiveness of protection via mix zones is dependent on their placement. Optimal placing of mix zones was studied in [106] and a placement optimization algorithm was proposed.

### 2.4.3 Obfuscation via False Locations

The addition of false locations can make it difficult for an adversary to find the exact location of a user. These approaches do not aim to hide users' identities instead relying on confusion as the main strategy. Reporting false locations does not require modifications to the LBS or the use of a trusted server although there is an added cost in communications. The advantage of these approaches is that they can be applied using real LBSs.

This was first explored in [107] where the authors attempt to make realistic false locations by ensuring false locations are in the same region as the original and cannot be further from the possible travel distance of a user. The authors also suggest a method to reduce extra communications by splitting positional data into two separate buckets that the server can recognize. This reduces the cost of communication from $O(n)$ to $O(log(n))$ where $n$ is the number of reports sent in communications to the LBS.

The authors in [108, 109] present an alternative strategy, false locations are incrementally increased and sent to the LBS until the query based on the actual location can be satisfied with the aid of local processing. The authors introduce the concepts of an anchor, supply space and demand space. An anchor is a fake location centered in the supply space and the demand space is the minimum space required to guarantee the query is satisfied. Queries are sent requesting POIs with increasing distance until the result is returned. However, this approach incurs high communications and processing overhead.

Krumm aimed to find a solution to an inference attack he presented in [87]. The identity of several pseudonymous were revealed in the Microsoft Geolife dataset. Two dimensional Gaussian noise was added with a standard deviation between 50m to 5km to each geospatial coordinate [87]. Interestingly, the amount of noise needed to prevent an attack from being successful was high with the standard deviation needing to be set to 5km in order to prevent the attack. This level of noise would likely render the dataset not usable for data mining purposes.

Controlled regions are used in [110] with the authors proposing a circle algorithm and grid algorithm. The circle algorithm centers the user position in the center and determines random positions within the circle while the grid algorithm overlays a uniform grid of $k$ points where one of those positions is the actual user location.

Extensions to random noise approaches was presented in [111] with the authors proposing three variations including N-Rand, N-Mix and N-Dispersion. The technique of adding Gaussian noise [87] was improved by introducing an $n$ parameter that does not select the first randomly generated point but rather the farthest point after $n$ trials. The idea of preserving $k$-Anonymity by using an overlay network to select a random node to initiate a query was improved by taking the average position of the 4 closest points to be the location of the initiator. This resolves the issue of being able to infer a location accurately when a large number of nodes are available. Dispersion involves selecting a random point at the edge of a circular region of radius $r$ then performing the operation again with a smaller circle to de-center the original location. This was improved by generating a random point within the entire area of the smaller circle.

A more recent approach presented a framework for achieving differential privacy guarantees by adding Laplace noise to geospatial coordinates in [112]. The authors formalize geo-indistinguishable, the main idea is that noise is added within a radius of each user to ensure each user is differentially private.

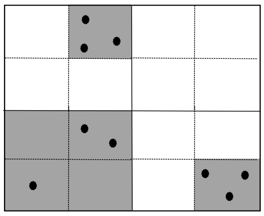


Figure 2.13: Quadtree Preserving 3-Anonymity

### 2.4.4 Obfuscation via Spatial Cloaking

Spatial cloaking techniques aim to use a region to blur the exact location of individuals. The size and shape of the region dependent on the privacy requirements of a user. As opposed to using a pseudonym and reporting the exact location, a user would send a region that contains $k$ people so they cannot be distinguished from $k - 1$ other people. This works well in areas that contain a high density of people.

Most techniques make use of either $k$-Anonymity or $l$-Diversity. The principle of $k$-Anonymity is referenced often in papers proposing cloaking techniques and was discussed in Section 2.3.2. Spatial $k$-Anonymity protection models ensure each user location is considered to be $k$-Anonymous [113] within a region. Modified queries are sent to the LBS to ensure that the LBSs cannot join data to a great accuracy. Spatial cloaking has been demonstrated to be effective in both centralized and decentralized architectures. An extensive study of the performance of common spatial cloaking algorithms are available in [114].

Adaptive interval cloaking was proposed in [115] by Gruteser. The aim is to ensure the region a user is located in overlaps with other individuals in the same region. The space is recursively divided using a QuadTree until each region satisfies the minimal value of $k$. It is assumed that each user would be satisfied with the same $k$-Anonymity requirement. Unfortunately this approach is not scalable as the tree handles the movements of each user individually requiring the tree to be constantly regenerated. Figure 2.13 shows how each shaded region is 3-anonymous, the region size is increased to enclose at least 3 points. This approach was later extended by Bettini [72] to take into account historical data. The authors noted that linkage attacks could be performed if patterns could be identified in data such as frequently visiting a restaurant or catching public transportation.

Gruteser also proposed an approach that considers $k$ to be the number of indistinguishable sensitive areas [116]. Sensitive areas are POIs such as restaurants and gas stations. The space is divided up so that each zone contains at least $k$ sensitive places. Users traverse from one zone to another when location is reported. Another approach considering sensitive areas was demonstrated in [98] that considers both an uncertainty region and a sensitive region such as a hospital. An uncertainty region is of a specified size and hides the user within a region. However, if the user is in a sensitive area then stricter privacy controls are enforced.

Users may want to be $k$-Anonymous but also specify a minimum area where they want to hide. CliqueClock [117] aims to allow users to set their desired level of privacy preferences. Each user specifies the level of $k$-Anonymity they desire and the maximum size of a cloaking region. Constraints of each user are captured by the graph data structure. Nodes are considered neighbors if and only if each of them is inside the others' constraint area. Cliques are considered to be a set of users forming the graph that are indistinguishable from one another. These regions are sent to the LBS when a query is initiated by a user.

Hilbert space filling curves were applied in [118] demonstrating significant savings in processing costs. The Hilbert order is determined for each user location and used as a means of bucketing users and calculating the minimum bounding rectangle for users within a range. Another approach also presented in [118] made use of nearest neighbors. First the set $S$ was found containing the user $u$ and $k-1$ of $u$'s nearest neighbors. Secondly a neighbor $v$ was selected from $S$. This process was repeated for $v$ to find the set $S'$ containing $v$'s $k-1$ nearest neighbors. The cloaked spatial region is the minimum bounding rectangle between users in $S'$ and $u$.

Casper proposed a novel query processor that can efficiently support users changing $k$-Anonymity and the minimal spatial area required instantly [101]. The use of a pyramid data structure as shown in Figure 2.14 is proposed that divides the space into grids at different levels of granularity. Grid cells dynamically maintain how many users are in that cell. It is important to note that only pyramid cells that could contain potentially cloaked regions are maintained. The spatial cloaking algorithm makes use of the pyramid structure. Starting from the bottom of a pyramid, the grid cell a user is located in is found. The cell is returned if it is large enough to satisfy the user's requirements, otherwise the algorithm traverses up the period until it finds a satisfactory region.

Mobile specific approaches have also been proposed focusing on masking the raw location by adding a distance error to a point and using rectangles [110, 119]. Ad-hoc mobile networks such as GSM lend themselves well to delegating certain nodes to act as a query

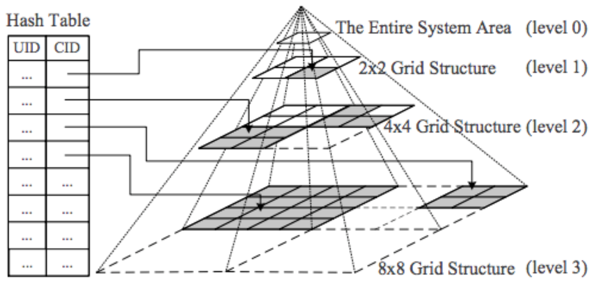



Figure 2.14: Casper's Pyramid Data Structure [101]

initiator on behalf of other nodes. This idea was proposed in [120] with the authors demonstrating how $k$-Anonymity can be achieved by having $k$ nodes use a particular node as a query initiator so users are distinguishable by $k - 1$ other users.

### 2.4.5 Location Based Service Architectures

The design decisions on the architecture for location privacy systems are impacted from protection strategies required to be implemented. Early research focused on centralized protection methods that involved the use of a trusted server [120]. Later methods shifted to decentralized paradigms removing the need of a trusted server due to the possibility of it being compromised by unauthorized access.

#### Centralized Models

Centralized approaches tend to be more practical for organizations to implement as they have more control over the system. A trusted anonymity server is accessed on behalf of the users [117, 105]. Typically algorithm functions using principles such as $k$-Anonymity are preserved before passing on the data to a LBSs. All identifying information including usernames or IP addresses are replaced with a pseudonym. Location points are spatially cloaked to ensure some level of privacy is preserved.

Centralized systems can also be detrimental to protecting user privacy if the trusted anonymity server is in some way compromised. For example if an adversary successfully installs malware on the anonymity server then protected queries can be sent directly to the adversary before nullifying any protection mechanisms. Therefore, an anonymity server can potentially become a valued source of location information for an adversary.

### Decentralized Models

Decentralized methods anonymize locations without the need of a centralized server. Computations to preserve privacy distributed using a peer-to-peer approach. Each node in the computation does not have knowledge of each user's movements. Users can feel more confident that they do not need to place their trust in an anonymity server.

The drawback is the lack of centralized control makes these systems difficult and costly to implement. Maintaining these types of systems can also be problematic. For example, if software needs requires updating, then every client also requires updating. In addition, there are overheads in bandwidth and computational power amongst peers that are participating which can lead to additional costs for both the users and owners of the system.

SpaceTwist is a simple example of a system that does not require a trusted anonymity server [108]. Multiple queries are incrementally sent at various locations to a query processing server until the query is satisfied. This makes it difficult for the server to identify which query is the actual query.

PRIVE relies on a decentralized architecture where any peer can act as a query initiator on behalf of another peer. The authors demonstrated in a simulation how mobile devices can organize themselves into a hierarchical overlay network [121]. Smartphones establish adhoc connections with each other using common wireless protocols such as WI-FI, GPRS or GSM. A server is still required to verify certificates used for encryption which is avoidable by having participants agree on an encryption scheme ahead of time. The user can set the degree of anonymity by specifying a minimum $k$ value. This value determines the number of clients required to perform a query the initiator is requesting. An adversary will have difficulty identifying the device in the overlay network that actually sent the query.

## 2.4.6 Query Protection for Specific Applications

Location privacy can also be achieved at query time depending on the type of application. A common function LBSs perform are nearest neighbor queries in a geographic space to find points of interest within proximity of the user.

For finding points of interest, an algorithm has been proposed that negotiates the level of accuracy in exchange for precision required when answering the query [75]. The authors propose an agent sends a pre-query that the server will reply with a number of POIs. If the agent agrees then the result is returned otherwise a smaller obfuscation set is requested. This allows for the agent to calibrate the privacy on the fly. Instead of sending a query to an LBS, the authors in [122] recommend a query contain code that is executed directly on the LBS server.

Generally most protected queries make use of cryptographic approaches. We also discuss Private Meetup [119] that is an application to organize optimal meeting locations without exposing user positions.

#### Cryptographic Methods for Location Query Protection

Location privacy can be protected via cryptographic approaches such as [123, 124, 125, 126] that only reveal location when certain conditions are met. These approaches do not require a complex architecture, however they incur additional processing and communications overheads. In addition, techniques used to improve efficiency of spatial processing such as indexes cannot be easily utilized.

A recommendation application of places a user may like to visit was demonstrated to provide quality results while privacy preserving in [123]. Recommendations can include suggestions for good restaurants or coffee shops. The system performs aggregate calculations of recommendations by spreading the load over multiple nodes in the network where they only expose information that is needed to assist in the calculation. In this manner, no nodes in the network have all the required information to locate users.

Nearest neighbor queries have been demonstrated to be performed using cryptographic methods in [124]. The one-way function of Hilbert curve transformations are used to encrypt two dimensional space to one dimensional space. An encryption key is supplied by a trusted entity that is then used by the user to encrypt their query, this query is then sent to the LBS. The LBS will evaluate the encrypted query and send the result back to the user in encrypted form in which the user can decrypt the result with key provided by the trusted entity.

A common application of LBSs is to provide a user with friends nearby. Using techniques from cryptography, guarantees were created to only reveal a location if a friend is nearby [125]. Three protocols are introduced: Louis, Lester and Pierre that achieve this goal in different ways.

Private information retrieval protocols (PIRs) were adapted in [126] to solve NN queries. The space is divided up into a grid with all objects stored in a format compatible with PIR.

The user only needs to provide the encrypted cell and the server will respond with nearest object. This process requires higher computational overhead because of added processing and communications.

### Private Meetup

PrivateMeetup uses crowdsourcing to organize meetings amongst users in a privacy preserving manner [119]. The use of crowdsourcing has gained popularity and is employed to solve a variety of problems by enlisting the assistance of a wide community. This approach has been particularly successful in the modeling community.

The algorithm used by PrivateMeetup uses crowdsourcing to protect user privacy by removing the role of a LBS and relies on shared data. A group nearest neighbor (GNN) query is introduced that allows a group to meet at their nearest point of interest (POI) such as a restaurant. The resulting meeting point minimizes the total or maximum distance of the group.

Users of PrivateMeetup are connected via the Internet using standard cellular networks or WI-FI. An initiator amongst a group of users initiates a MeetUp request and proposes a set of POIs Users can then choose to accept or deny the request from the initiator. Those participating then also propose a set of candidate POIs. A GNN query is then performed using the set of candidate POIs to find a suitable meeting location. The focus of the private meeting algorithm is to perform a $k$-GNN query where all users hide their exact locations.

In decentralized models like PrivateMeetup, users that process the $k$-GNN query could be considered as potential adversaries. For this reason the algorithm is designed so each user has the same amount of knowledge as a potential eavesdropper. Wireless ad-hoc networks should not be used for this approach to prevent a range based attack. It is assumed that each of the group members do not know each other and no adversary has background knowledge about any of the users. The location of surrounding POIs are also not revealed to prevent a distance based intersection attack.

Location points are commonly represented using 2D space. Points can be converted to a 1-dimensional imprecise distance space with distances ranging from 0 to the maximum possible distance between any two locations in the entire Euclidean space. Buckets are derived by iteratively dividing the range of distances. Users only are required to reveal the bucket in which the actual distance to a POI is within. Smaller buckets represents higher imprecision and thus greater privacy. The algorithm initially starts using a high level of imprecision and iteratively reduces the imprecision until a meeting place is decided upon.

An initiator creates a MeetUp request and selects a group of users they want to meet. The users that accept select POIs as potential meeting places. The initiator receives all POIs and computes imprecise distance space. Each member including the initiator computes their distances and update the imprecise aggregate distance for each POI provided the user privacy levels are respected. When the initiator receives the returned data set, POIs that cannot be GNNs are discarded. Given that the data set size equals $k$ then locations are announced to the group otherwise another round of refinement is performed.

### 2.4.7 Location Privacy vs. Utility Trade-off

It has been well-established in location privacy literature that utility is lost in exchange for privacy. A user that wishes to enjoy a high degree of privacy loses the benefit of receiving high quality query results. Ideally, the user would have the desire to set the privacy preference to balance they are comfortable with.

Every user has different privacy preferences and requirements, studies have been performed aiming to work out how to best present privacy settings. In a user study conducted in [127] to evaluate the location privacy preferences of 19 people, it was found that users set really complicated privacy profiles. Through the use of extensive surveys, it was found in [128] that a few categorizations could capture all user preferences. A similar study also found this in [129].

Depending on the protection mechanism used, changes how the privacy and utility can be balanced. Additional false locations can be added when using the false reporting strategy to confuse potential adversaries. The time frequency of how often to switch pseudonyms can be set when using mix zones. Obviously with spatial cloaking, user specific values for $k$ and the minimum size of the cloaked region can be set.

While many studies evaluate how private the proposed approaches are, few actually consider in detail the utility of the results. It has been suggested in [130] that the dataset of actual results if no protection methods were used versus privacy preserved results should be measured in terms of distance. This allows a score to be assigned to evaluate the utility of an approach. A good approach would yield a balance between privacy and quality of results.

### 2.4.8 Measuring Location Privacy

Location privacy as a concept has been well defined with formal definitions available that share common and consistent themes. However, how to measure it yet to be standardized making it difficult to compare location privacy protection methods. One general trend observed across different measures is the quality of the LBS responses decreases as location privacy increases [74, 120, 131].

Authors have suggested the amount of false locations as a discrete measure. The *level of privacy* measure has been defined as the number of location coordinates sent to a LBS in a single query in [74]. The authors achieve ambiguity by sending more points to a system, thus making it harder for an LBS to determine the actual user location. This allows the system to still supply the correct answer to the user. The issue with this method is that if points are sent from a remote area, there may be only one location that makes sense when viewing on a map. Using a similar principle, approaches that use $k$-Anonymity such as [115] defined higher values of $k$ to represent greater levels of privacy. This measure does not consider an adversary may have access to background information that can effectively reduce $k$ to few levels that make it trivial to infer the actual location.

The distance between a user's actual location and the expected distance between the best possible estimate an attack can infer has also been proposed [79]. This measure requires that possible attacks are well defined which is hard to achieve in practice as all possible attacks are not known. The primary author in a follow up study in [131] evaluated their method by defining location piracy as the duration that an adversary can track a trajectory. Probabilities have also been suggested in [104] where behavioral probabilities are attached to certain events. For example, a user is less likely to perform a U-turn. This allows for an adversary to narrow down a user's path.

The common trend in evaluating these suggestions is that it is difficult to find multiple studies that measure location privacy in the same manner. The ability to compare and evaluate the effectiveness of techniques is important for the progression of the field, in addition to making it easier for location services providers to adopt and test protection methods. Measures have been either discrete or continuous and each have their drawbacks.

Location privacy is difficult to measure, there is no single measure that covers all possible scenarios. As the field matures and more measures emerge, there will be eventually a standardized set that would be expected to be evaluated when presenting novel protection techniques. This has already started to occur with a framework presented in [132]. The framework presents formal definitions for many inference attacks presented in the literature and how to apply the framework to evaluate privacy preserving methods.

### 2.4.9 User Concern for Location Privacy

Privacy researchers consider the preservation of location privacy as an important area. It was argued in early research that techniques are important in order for the public to feel comfortable in adopting new location technologies. Surveys into user attitudes seem to indicate that there is not much concern. However, it is suspected, this lack of concern can possibly be attributed to users not fully comprehending the consequences.

We note that most research into user attitudes into location privacy were conducted before smartphones proliferated and the general public was not as aware of location tracking. There is no definitive study on user attitudes but there is evidence to suggest that being aware of the negative consequences can lead to user concern.

Barkhuus et al. studied privacy attitudes to LBS services [28]. Four functions were made available to participants including a ringing profile automatically set for private settings, a ringing profile automatically set in public settings, a service to suggest lunch when the user is within range of a restaurant and an alert for when friends are within proximity. The study included 16 participants aged between 19 and 35. Interviews conducted found that position-aware services are less intrusive then location-tracking services. Participants ranked privacy concern from a scale of 1 to 5 with 5 being *"highly concerned"*. The average score was low being 2.75, however the study did not distinguish about awareness of consequences.

User attitudes towards location aware services was explored in [133] in 2003. Future possibilities of personal navigation products was presented to 55 participants of diverse demographics during a trade fair in Finland. Participants indicated that they are happy with information sent to them provided it was relevant to the situation. In the group interviews, some people were worried about privacy indicating that it is like *"big brother"*. The participants that did understand tracking was possible were not too concerned about the privacy issues in regard to the network operators indicating that they already trust them with sustainable personal information. The users put a great deal of faith in the operators and also suggested regulations prevent abuse. The authors suggested that any abuse of personal data can lead to the public loosing trust in services that is hard to win back.

Subsequent case studies performed by Barkhuus found that users are initially concerned with location privacy but discontinue to have concerns once using location based services [134]. The initial case study had 23 participants and they had to imagine using an *"imaginary"* LBS with additional services added throughout the week. A later case study made available a real LBS to staff and students on a campus environment at the University of California. Data was gathered via a questionnaire with 35 students' participants and a quantitative study of 12 students. Users expressed concern and were surprised by the amount of data transmitted to services. One female participant felt her privacy was invaded when a peer commented how was her gym session when tagging herself at the gym in the study when another participant saw it. Users that did not feel they benefited from the features indicated more concern than those that found them useful. The authors suggested that privacy expectations and functionality provided are required to be balanced.

Iachello tested user attitudes towards location aware apps back in 2005 before they gained mainstream adoption [135]. Reno is a peer-to-peer location based messaging service deployed in a two-week study to better understand how people would use these types of services [135]. Participants in the study were two families with teenage children in Seattle. In total there were 11 participants. Reno requires users to define place names such as *"Home"* or *"School"* and then define where they are located by sending a location sample. The study found that privacy concerns were balanced with broader requirements including control and application utility. All participants said they would not share their location to anyone who asked but did not mind sharing with acquaintances.

The price individuals attributed to their location privacy has been investigated [136, 137]. University students surveyed were relaxed about sharing location privacy [136] and are even more likely for payment [137]. A study employed experimental techniques to determine the amount required to persuade individuals to reveal precise location information [137]. A fictitious auction was setup asking students how much they are willing to bid to be compensated for supplying location data given there is only a limited number of participants. For just 10 pounds, 74 undergraduate students form UCL were willing to give away their location tracks and 20 pounds if being used for commercial purposes. Although the results did indicate that whether the student travels outside campus or not is a consideration as well as the number of personal responsibilities.

A similar follow up study to [137] about the value of location privacy data was conducted in [138] that considered a more diverse demographic. In the study, 1200 people from five countries in Europe were being surveyed. It was found that different countries place attribute different values for privacy with Greece showing the most caution and placing the highest bid values. Gender differences were not apparent for sharing one month of data but women expressed more caution when asked to share a year's worth of data. The frequency of samples made a substantial different to bids. Participants placed a higher value when samples were captured more frequently. Overall, the findings in the initial study [137] that users are willing to give away 1 month of location data for 20 pounds for commercial use.

## 2.5 Trajectory Privacy

Navigation tasks traditionally performed by in-car navigation systems are being commonly replaced by smartphones leading to smartphones not only capturing locations, but travelled routes as well. Trajectory privacy is a relatively new research discipline that extends from location privacy taking into account the spatial temporal nature of individual location points. While location privacy focuses on keeping individual locations private, trajectory privacy aims to protect from information being inferred across an entire trip.

Mining trajectories makes it possible to identify places important to a user, provide personalized recommendations and determine relations between users [139, 140]. These functions usually provide some kind of benefit for users of LBS services such as turn-by-turn navigation and finding routes with least traffic congestion. Researchers aim to perform these functions while maintaining user trajectory privacy.

Proposed protection techniques are based on either trajectory confusion, generalization and suppression-based methods or trajectory $k$-Anonymity. These methods for trajectory protection are covered in this section.

### 2.5.1 Trajectory Confusion

Trajectories have specific characteristics that make them unique in comparison to others. For example, trajectories that contain a main road or highway are more likely to be similar to trajectories that only contain residential streets. Perturbing these unique characteristics from paths can be used as a method or protection. The added benefit is that trajectory mining algorithms will still produce comparable results without violating user trajectory privacy.

An issue was noted in evaluating a tracking algorithm finding that it was difficult to infer the identity of anonymous location tracks of multiple users where there paths cross [77]. This weakness was exploited in [79] as a protection mechanism where location tracks were perturbed to make it difficult to tell if the tracks touched or not. This idea was extended to include false locations around where the tracks intersect [141]. The temporal dimension can be degraded as well instead of just obfuscating over space by removing location samples. Reducing location samples reduces the success of attacks [93, 131].

Inserting dummy trajectories has been proposed as a method to protect against adversaries inferring information from trajectories in [141]. The authors propose two schemes being random pattern scheme and rotation pattern scheme to achieve this. Parameters specified in a user profile include short-term disclosure, long-term disclosure and distance deviation thresholds. The motivation for using dummies is that a trusted server or infrastructure does not need to be relied upon. Movement patterns have been demonstrated to be unique, so a key challenge is to generate realistic dummy paths.

The space into a grid to simplify the approach and assumed that users were free to move in the space. The random pattern scheme takes the starting point and requires the destination point be known ahead of time. A dummy can move vertically, horizontally or both based on the speed. The dummy path moves randomly from the starting point to the destination. Provided that dummy paths also produce consistent movement patterns, it would be hard for an adversary to identify the truth path. Including more dummy paths strengthens privacy but adds computational overhead.

The rotation pattern scheme is more sophisticated than the random scheme and ensures dummy paths significantly deviate. The key aim of the scheme is to ensure there are intersections present within the actual user and dummy trajectories. Dummy trajectories are generated by rotating the actual user trajectory. The intersection point between the actual trajectory and the dummy is the rotation point. The scheme is required to satisfy the user privacy profile. The solution space is first generated to ensure distance derivation is satisfied. Then the short-term and long-term disclosures are obtained within the solution space. All trajectories found with disclosures less than what is specified are selected as dummy trajectories.

Results in [141] indicated that generating random paths with consistent movements was shown to improve privacy for long term disclosure of data. The rotational scheme required a similar amount of dummies to protect short term disclosure but significantly less paths to protect long term disclosure.

### 2.5.2 Generalization and Suppression-Based Method

These methods work under the assumption that adversaries may have different and incomplete trajectory information of users'. The aim is to reduce the probability of reconstructing the entire trajectory with the partial information. Quasi-identifiers are considered projections of trajectories that an adversary can access.

An anonymization algorithm that takes inspiration from the concept of generalization can be applied to preserve trajectory privacy [142]. The key idea behind the proposed approach is to take long trajectories and break it up smaller and simplified trajectories. This allows for increased projections and diversifies locations making it difficult to infer with high certainty what points comprise a trajectory. This is achieved by suppressing the existence of certain points within a trajectory by considering how omitting the point will alter the main direction and what is the benefit in terms of privacy.

Determining the optimal set of points to remove is considered an NP-hard problem [58]. Therefore, the authors propose a greedy algorithm continues to suppress locations until privacy is met. An attack is performed at each iteration of suppression until a privacy constraint is met.

C-Safety takes a different approach aiming to constrain the probability of an adversary inferring a user is at a sensitive place [143]. The authors formalize privacy-violating inferences and provide an upper bound $c$ to determining if a user is at a sensitive location. An algorithm is proposed that transforms a set of trajectories to a c-safe counterpart which can be published safely.

### 2.5.3 Trajectory $k$-Anonymity

The main concept underpinning trajectory $k$-Anonymity is to guarantee that trajectories are indistinguishable from $k - 1$ trajectories in a dataset. This is a deviation from location $k$-Anonymity as an entire trip is considered and not just a single location point.

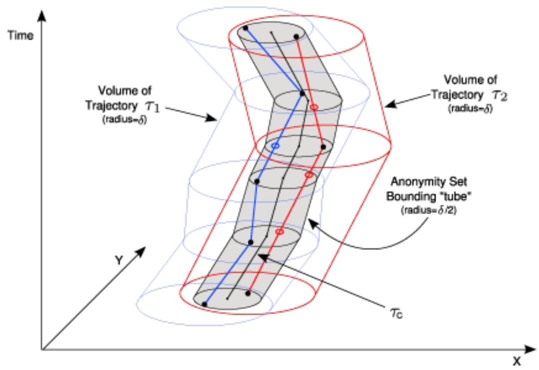



Figure 2.15: Never Walk Alone [144]

A clustering based approach was presented in [144] and illustrated in Figure 2.15 makes use of uncertainty in GPS measurements and uncertainty in multiple trajectories being co-localized within the same period to aggregate a $k$-animized trajectory. The authors define $(k, \delta)$ anonymity to ensure that a trajectory cannot be distinguished from at least $k - 1$ other trajectories bounded by the volume of the cylinder $(\delta)$. The set of $k$ co-localized trajectories is referred to as the anonymity set of $k$ trajectory. A clustering based approach is proposed to derive the anonymity set of $k$. A clustering approach is proposed to achieve this type of privacy that includes consists of three main phases being the pre-processing phase, clustering phase and space transformation phase. In the pre-processing phase, all trajectories with a similar start and end time are grouped in the same equivalence class. In the clustering phase, a greedy scheme is used to determine pivot points for trajectory. Finally, the space transformation phase aggregates each trajectory by moving samples within a cluster to the cluster center point.

Conversely, it was argued by the authors of [145] that enforcing classical $k$-Anonymity is not sufficient. The reasoning is because a fixed set of quasi-identifier attributes does not exist for all trajectories. Instead, they propose approaches referred to as extreme-union and symmetric anonymization. Alternatively, instead of representing a trajectory in a relational database, graph based approaches have also been demonstrated that partition trajectories based on similarity [146]. The authors of [147] demonstrate how trajectory privacy can be preserved by extracting the significant stay points and protecting them.

## 2.6 Privacy Preserving Data Mining

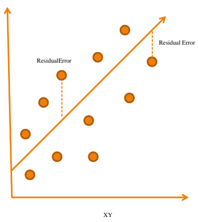
The personal information contained in smartphones is of value to researchers and analysts in testing a hypothesis or finding useful insights. This data can be mined via smartphone apps and processed using data mining algorithms. In this section, we present techniques that can be used without exposing user data. These approaches are current attempts to perform data mining while taking user privacy into consideration. Thus, they can be implemented on user smartphones to train models without the user exposing their sensitive information.

The most common supervised learning techniques used are linear regression, neural networks, support-vector machines (SVM), decision tree and random forest. Self-organizing maps is another unsupervised method useful to find patterns in data that are hard to detect with other methods. These techniques have practical applications across a range of disciplines. Literature on how to apply these methods while preserving privacy is available although limited. Despite there being much research into the techniques themselves, actual tools for production use to perform these methods privately are not widely available.

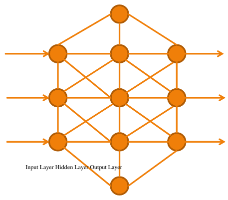
Linear regression is a modeling technique that attempts to find the relationship between two variables by fitting a linear equation to the observed data. There are a variety of methods available to fitting a linear line, including least-squares regression, maximum likelihood estimation and Bayesian linear regression. Multiple variables can also be predicted via a modified version of linear regression known as multivariate linear regression. Once a linear model has been fit, residual testing can be performed to test the variance between the actual data and the models results. An assumption made using linear regression is the data has a linear relationship. As it is one of the most commonly used modeling techniques and has been studied extensively.

The authors in [148] demonstrate how to use S2C for linear regression. Consider two parties Alice and Bob that each have a private dataset that they want to share certain attributes to fit a linear model. Two protocols are proposed, one that uses a semi-trusted commodity server and another that communicates between the peers directly. The line is fitted using matrix calculations. Each party calculates their part of the matrix before sending it to the other party. Once the matrix calculation is complete, a line is fitted.

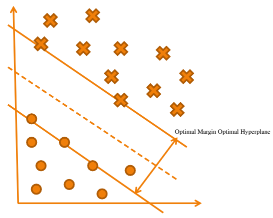
Neural networks are a supervised machine learning technique inspired from how the central nervous system and the brain works in biology. A neural network is made up of a



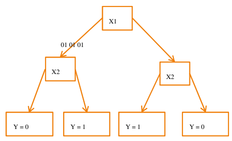

(a) Linear Regression



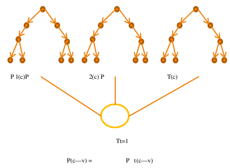

(b) Neural Networks




(c) Support Vector Machines




(d) Decision Tree



(e) Random Forest

Figure 2.16: Common Data Mining Approaches Used for Training and Classification

series of weighted and hidden neurons, a common training algorithm for neural networks is the back-propagation algorithm [149]. The back-propagation algorithm lends itself to be distributed and can be performed privately [150]. Neural networks can be trained across multiple machines containing different datasets without each machine knowing the contents of the data.

Support Vector Machines (SVM) is a machine learning technique based on the principal that you can define an optimal linear decision boundary in data [151]. The boundary the SVM produces makes a best attempt to define where one classification starts and another one ends, in order for SVM's to achieve high accuracy, the data set needs to be linearly separable. Given very complex training data, in practice it is highly unlikely classes would be linearly separable, slack variables were introduced to relax constraints.

SVM's combine three different ideas, finding the optimal hyperplanes, using soft margins to relax finding the solution to the optimal hyperplane problem and using convolution to map the feature space to vectors. The optimal hyperplane is a theorem defining the maximum margin between two vectors that can be perfectly separable. Privacy preserving SVM has been proposed in [152, 153]. The class labels for each variable are shared between each party that is holding data and jointly training the model.

Decision tree learning is an extensively used data mining technique to predict a target variable given a complex series of inputs [154]. Decision trees build of the concept of decision trees by creating a predictive model that traverses the tree. Many algorithms to build decision trees from data have been derived, the most common are ID3, C4.5, CART, CHAID, MARS and Conditional Inference Trees. Each algorithm builds a series of decisions used to segment the data into different subgroups. To illustrate a basic decision tree, below demonstrates how the XOR function could be captured.

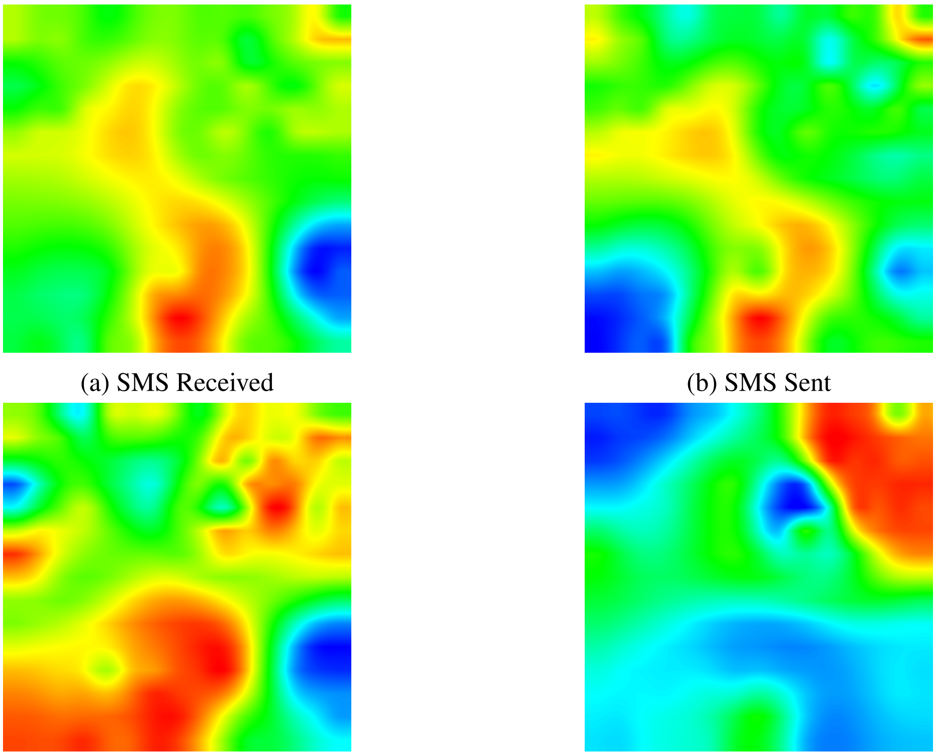
It has been proposed to build decision trees privately in [155] by applying Shamir's secret sharing scheme. The sum of secret values from multiple parties without revealing the secret values to each party. At the end of the computation, parties only knew its own value and the same of all the secret values but nothing else.

Machine learning technique extending of decision trees is the random forest [156]. It can predict the values of discrete or continuous variables and discover relationships between variables. It works by constructing a model consisting of decision trees based on the input training data and the classification variable. In comparison to unsupervised methods, it is more suited for targeting specific variables rather than data exploration.

Random forest has demonstrated good predictive power when the correct parameters are selected and sufficient trees are constructed. It can also provide good estimates of what variables are important for classification. Expanding of the concept of training a single basic decision trees by further training multiple decision trees to create a forest of trees. Each tree has the same distribution while a random variable determines the vector data to train each decision tree.

A method that utilizes deferential privacy has been proposed to train a random forest while maintaining high utility [157]. Each decision tree trained in the random forest ensemble is differentially private. When creating a decision tree, first an attribute is selected at a tree node that best splits the data. Child nodes are populated where the node was split and statistics are computed for the leaf nodes. The measure of information gain (IG) is used to determine attributes that are best for splitting. Laplacian noise is added to the rankings of IG based attributes and later to the computed statistics than generate the classification labels. The decision trees are ensembled in a manner that ensures the variance of each variable is unbiased. The results of this approach indicate that high utility is maintained when comparing to standard random forest.

The self organizing map (SOM) is an unsupervised neural network commonly used to explore complex data. Its key benefit is its ability to consider multiple data attributes at once by applying a mathematical model to all the available data. Two dimensional visual



(a) SMS Received (b) SMS Sent (c) Phone Ringing (d) Phone Signal GSM

Figure 2.17: SOM Images of Common Mobile Data Features

representations are generated where an analyst can draw inferences from the data. SOMs are suitable for accelerating the analysis in determining what key variables to consider. Examples of SOM images for smartphone data features are shown in Figure 2.17.

SOM's can be trained and evaluated while preserving privacy of the individual data [158]. SOM does not make use of any hidden layers and is a feed-forward network. The challenges of training a SOM securely is to discover the winner neurons privately and securely update the weight vectors of neurons and determine when the SOM training has been completed. Protocols have been proposed by the authors to perform these functions.

The method assumes that there is at most two parties training the SOM. Initially each neuron is initialized with its weight vector. Random input data is selected at each iteration of training. The distance between the input data and weight vector neuron is calculated and the winning neuron is found by running the *Secure Computation of Closest Cluster Protocol*. Weights are updated based on the winning neuron using the *Secure Weight Vector Update Protocol*. Both parties are updated on the progression of the SOM and training is continued until the *Secure Detection of Termination Protocol* halts the process.

Traditional clustering techniques are also available to perform segmentations that can be displayed on the visual representations. SOMs are a popular tool for conducting complex behavioral analytics. When reading a SOM visual it is important to understand that proximity on the map means that nodes are similar while taking all data into consideration. Once the model places an account in a particular node, it remains in that node on all visuals. This allows analysis by using the colors of the visual of all the data attributes. There are no axes, instead each of the maps scale drives the colors on the map, essentially, a heat-map.

The SOM maps below illustrate the relationship between aggregated variables from a mobile device. There is a clear relationship between SMS Received and SMS Sent, however there are nodes showing different behaviors in sections.

## 2.7 Smartphone Privacy

The architecture of smartphones influences how private stored user data is kept. General privacy concerns related to smartphone users discussed earlier were dataset privacy, location privacy and trajectory privacy. Initial research related to mobile privacy heavily focused on location privacy and considered it the number one concern. There is a now additional concerns due to the growing number of sensitive data types smartphones store as they become more sophisticated. Smartphones not only capture location tracks but also store other sensitive metrics like bank account information and photos of friends and family.

In this section, we focus on smartphone user attitudes towards privacy and comprehension of security and privacy mechanisms put in place to protect them. Modern smartphone platforms provide rich development environments allowing third party apps to access much of the devices native functionality. This functionality has been exploited to mine for user information. Bluetooth is a well-established wireless communications standard that is embedded into every modern smartphone. While Bluetooth provides great convenience, security and privacy concerns have been noted. We discuss how Bluetooth can be exploited to learn information about an individual.

### 2.7.1 Smartphone Malware

Smartphones being pervasive in society makes them a target of adversaries who wish to mine for sensitive information or control a device for other purposes such as running a distributed bot-net. In this section, we provide a brief history of the evolution of malware on mobile devices, how the malware is propagated and common intended purposes. Malware continues to be a growing problem for smartphones. Several investigations have attempted to document the evolution [159, 160, 161, 162].

Malware has different intended purposes as detailed by Felt et al [161]. The main aim of Malware is to gain access to a device to perform functions for adversary such as extracting data, causing the device to malfunction or causing issues for the user such as sending intrusive alerts. An adversary will find a way to directly install the malware using techniques such as social engineering or finding a vulnerability in the device to install the malware. Spyware is a class of Malware that sends information about a person such as messages they receive and calls they make. Spyware is a popular method used by suspicious spouses and while it is legal to download, it is illegal to deploy without the owner of a smartphones knowledge. Grayware is another specific class that provides a useful function to the user but also may perform malicious tasks in the background such as collect personal information.

Malicious malware can be delivered via multiple infection vectors. The most common include SMS/MMS, Bluetooth (Described in detail in Section 2.7.3), Internet access and USB [163]. Cellular services have long provided SMS services to send short 140 character messages and later MMS messages. SMS services do not require the user have access to the Internet and can be used to deliver malware if an exploit can be found or spread malware by sending links to all users in the victim's phonebook. Smartphones often maintain continuous connection to the Internet exposing it to the same threats as PCs. Browsing the Internet and downloading email attachments can lead to malware being deployed without the user's knowledge. USB can also be used via infecting synchronization services that move data between PCs and mobile devices. Another recent example is charging stations can be infected and spread malware when a user attempts to charge their device demonstrated in [164].

There are a number of risks posed by a user's smartphone being infected by malware [163]. The system can be damaged by draining the battery or make the system unusable by changing settings and disabling functionality such as calls. Economic loss is also possible through the use of dialing or messaging premium rate numbers without the user being aware or deleting important information such as documents or photos on the SD card. Another common use case is to establish a bot-net for denial service attacks of mining for crypto-currencies. Furthermore, infected nodes can deplete a networks resources by generating a high amount of fictitious traffic. Information leakage is also a concern though the use of malware extracting personal information about a user and is a major focus of this thesis.

While smartphone malware was discussed early on as a theoretical threat, actual cases of malware has been found to exist on all the popular mobile platforms including Symbian, Android, iOS, J2ME and windows mobile. Kaspersky lab discovered Cabir [165], the very first smartphone malware and propagated via Bluetooth. The first Android Trojan labeled Trojan-SMS.AndroidOS.FakePlayer.a [166] pretended to be a media player while running malicious code in the background to sending premium SMS messages to extort money from those that installed the app. Ikee [167] was the first worm targeting jail broken iPhones by using the installed SSH server to access the device. J2ME was the most targeted mobile platform up until 2011 when the market dominance shifted to Android and iOS. Jupiter networks noted in 2011 Android became the most targeted platform shifting from J2ME [168]. Windows Mobile has also been targeted with Brador [169], the malware opens up a backdoor between using a TCP/IP connection.

### 2.7.2 App Privacy

Research into the privacy implications of third-party app usage is a relatively new topic. Automated systems have been proposed for both iOS and Android to detect privacy leaks at an API method call level. While these systems are very useful in detecting where sensitive information is leaked, they do not indicate if it is justified given the functionality of the app.

The iOS platform relies on a strict auditing process to protect their users as opposed to a permissions system. A system called PiOS [170] gained much attention demonstrating the ability to de-construct an iOS application and demonstrate where privacy leaks occur. PiOS found that most of the applications that were analyzed on iOS do not leak much personal information, however more than half leaked the device ID. Considering Apple perform many quality checks on apps during submission and are quick to remove any apps that malicious apps that make it past the submission process, the results make sense. It should be noted however that users that jailbreak their phones and install apps through non-official sources may be at more risk since there is no auditing process or permissions system.

A similar a system called AndroidLeaks [171] was created for the Android platform also capable of automatically detecting potential privacy leaks. By decompiling the Java byte code, AndroidLeaks analyzes an app by looking for methods that pass personal information to the third-party app. In the case of Android, many applications were found to be leaking private information.

Another system called Stowaway was demonstrated in [9] which compares method calls made by an app relative to the permissions the app developers request in the Android Manifest. It was found that one third of applications are not following the least privilege path with their permission requests. For example, a developer may request fine-grained location access but only actually use coarse grained access. This is likely a result of developers not understanding how to request permissions correctly due to unclear documentation.

Currently there is nothing in the literature that allow for method calls to private information were justified based on the advertised functionality of the app. A location based social network will need access to location services and a messaging app will need access to a user's contact. When an app is requesting information that is not required, that is when a user has more cause to be concerned as there is no motivation for the information other than to store it.

### 2.7.3 Bluetooth Privacy

Bluetooth is a wireless standard used to facilitate devices sending data over short distances. Its intended application was to be used as the communications layer in wireless headsets. Applications for Bluetooth have since expanded considerably to include speakers and file transfer. Almost every mobile from released in from the year 2000 onwards are Bluetooth equipped which have made it a target for hackers. Bluetooth has been of concern to privacy researchers as several vulnerabilities have been identified allowing for an attacker to turn on the microphone, extract contacts data and upload malware.

Nine attack vectors were proposed in a Bluetooth Threat Taxonomy [172]. These vectors are Surveillance, Range extension, Obfuscation, Fuzzer, Sniffing, Denial of Service, Malware, Unauthorized direct data access and man in the middle attacks. In this section, we provide a summary of each of these Bluetooth attack vectors with an expanded focus on how these can be exploited to invade a user's privacy.

**Bluetooth Surveillance** Bluetooth surveillance techniques aim to profile a device in range in order to gather information that may be useful for an adversary. Types of information include device specifications, communications channels and physical location for the purposes of location tracking. It is also useful to know what services the device supports such as headset communication or file transfer. Surveillance techniques do not pose any threat to the device but do yield potential insights about a user that are considered privacy invasive.

**Bluetooth Range Extension** Each Bluetooth device has a uniquely assigned address which provides key information about the specifications of the device. An adversary obtaining device information could then look up known exploits in order to gain access to the device. Any device in discoverable mode broadcasts its address. Even devices not in discoverable mode can probed to determine if they are in range if the address is known using specialized hardware and software. A smartphone a user owners is indirectly provided to surrounding years.

**Bluetooth Obfuscation** Communications channels can also be probed to determine what services a device supports which can lead to an adversary gaining unauthorized access. Bluetooth communicates using RFCOMM channels which is similar to TCP/IP. A device can be scanned for services that are not secure. Even gaining access to one unsecured service can lead to a privacy invasion. For example, an adversary can ears drop on conversations with access to the audio headset service.

**Bluetooth Fuzzer** Location tracking can also be performing using surveillance techniques. Specialized Bluetooth beacons can be deployed in areas that are of interest to determine if persons of interest are in the area. A technique referred to as War Nimbling has been demonstrated to be taken further by equipping beacons with camera. Whenever a user is in proximity, the camera takes a photo in the direction they estimate the device is whenever they walk past the beacon. In Chapter 4, we demonstrate how Bluetooth can be used to locate users indoors and propose a localization scheme.

**Bluetooth Range Extension** Bluetooth works under the assumption that communication is short range, normally within 10m. Short distances of communication can be overcome with high range antennas that allow for communications from kilometers away. An attack can be performed from a discreet distance allowing the adversary to access private information without exposing themselves. A user may feel it is safe to leave Bluetooth on because there is nobody in the surrounding area or they are not in a public location such as apartment building. This assumption is incorrect as modern smartphones do not provide any indicator that Bluetooth communications is occurring. There is no network monitor such as a flashing icon that indicates data is being sent and received. Thus, the user cannot detect when devices are accessing a device unless they have the technical sophistication to understand how to access the system logs.

**Bluetooth Range extension** Bluetooth devices are all assigned a unique address, normally be the chipset manufacturer. Tools have been developed that allow for an adversary to change the identity of their Bluetooth device to that of another device. Thus, it cannot be assumed that the device that is being communicated with is actually the trusted device initially paired. The trusted device may initially connect to a device, a device with the same identity can then intercept the packets that are sent between the devices. An adversary may be able to learn information from the packets intercepted.

**Bluetooth Fuzzer** Fuzzing tests how the Bluetooth stack of a device handles various inputs to try determine any flaws in how the stack is programmed. Similar to TCP/IP packets, Bluetooth packets follow a well-defined formatting standard. Assumptions are made on the length of certain fields within the packet. While information cannot be obtained about a smartphone user directly from fuzzing, it can be used to cause inconvenience to a user.

Various custom Bluetooth packets can be constructed by an adversary and sent to a device. One possibility of sending a malformed packet is it could cause the Bluetooth stack to crash. Assuming a smartphone user is on an important call and using a headset, the call will be interrupted. This could have unintended consequences for the user if the person they were on the phone to assumes they hung up on them. Another possibility is the entire smartphone can crash since the Bluetooth stack normally runs at kernel level. Considering that smartphone users are now performing office tasks on a device, important work may be lost. Causing inconvenience to a user is a form of privacy invasion. As stated, even if information is not obtained, the user is still disrupted from usual activities they perform on their smartphones. Bluetooth communicating over radio frequency channels is susceptible to packet sniffing. Commercial packet sniffers are available for purchase that intercept data in range. Packets intercepted normally have encrypted contents, although analyzing the packet structure itself can be used to discover vulnerabilities.

Encryption of the contents of the packet leads to the assumption that others cannot eavesdrop on communications or intercept files. However, if the encryption scheme is compromised on the devices then the contents can be decrypted. An example is if two smartphones have installed a custom public/private certificate which apps then use to encrypt the data. An adversary can then decrypt the contents of the packet given they have access to the private key. Information about a smartphone user could be learned simply by monitoring Bluetooth traffic over a period of time.

**Bluetooth DDos** Communications platforms are susceptible to denial of service attacks. These attacks involve sending requests at a high frequently to saturate the devices resources. Radio technologies like Bluetooth are susceptible to jamming and constant requests can be used as a form of abuse. When sending a request to a device, it often prompts the user. These prompts can be used to send unsolicited messages or render the device unusable. The right to privacy is also the right to be left alone. Flooding a device a user uses frequently is a form of privacy invasion.

**Bluetooth Malware** Bluetooth has been used as a method to spread malware by attacks. Normally a security flaw with the Bluetooth stack is required in order to deploy the malware. Attacks have been demonstrated in the past where a smartphone user is asked to accept a file. Once pressing accept, the malware is installed on the device without the user being aware. Bluetooth is not used to install apps so these types of attacks are rare. However, if executed, the malware could spread really fast given how almost every person has a smartphone. Installed malware will be able to access all a user's sensitive information on the device as well as monitor the smartphone user without them being aware.

It is possible to gain full access to a device via the Bluetooth communications channel using security flaws or loop holes. One such example is an attempt to Bruteforce the pin of a device. The time it takes to break a pin is often proportional to the length of the pin. It has been demonstrated that a four digit pin can be broken in minutes, however a 16 digit pin can take thousands of years with current technology. Many smartphone users do not even change the default pin, often attempting to connect with manufacture defaults work. Once a device is paired then every service provided by the Bluetooth channel is accessible. Using additional security flaws an adversary may be able to execute arbitrary code and gain full access to the device.

These attacks involve a device acting as a relay between two devices that actually want to communicate. By acting as a relay a packet is intercepted and sent to its intended destination. This type of attack may allow for an encryption key to be learned which will then allow sniffing tools to decrypt the data it intercepts. Many of these attacks have been prevented with recent security upgrades. However, authentication models such as just works are still susceptible. In public areas such as an office complex where many Bluetooth devices are operating, it is possible that a user connects to the relay with a similar name as the intended device instead making these attacks possible.

### 2.7.4 User Comprehension of Smartphone Privacy Controls

There has been research conducted to better understand the level of privacy comprehension of smartphone users. Surveys indicate that user comprehension is low which inhibits them to make the correct privacy minded decisions. However, users do express actual concern for their digital privacy.

Early work performed by Chin et al. into understanding user attitudes towards smartphones found that users were hesitant to complete certain tasks [173]. The aim was to determine if users avoid smartphones due to privacy concerns verses their privacy attitudes towards traditional PCs. It also aimed to find insights into how users evaluate what apps they consider trustworthy. Interviews and surveys were conducted for 60 users across various demographics to eliminate potential sources of bias.

Chin et al. discovered that users' have an equal amount of concern for security both on smartphones and laptops [173]. Users were more concerned about privacy on mobile devices since smartphones contain more personal information. Users that were more concerned about privacy on their laptop had the opposite response saying they have more personal information on their laptop. The amount of sensitive data stored on a device indicates a user's level of concern. Some users perceptions of security were influenced from incorrect knowledge about wireless technology. Participants felt wireless networks such as GSM/WI-FI were less secure and be easily intercepted.

It was also found in [173] that users are worried about physical phone loss or damage because of all the sensitive data it contains like photos of friends and family. In the study, 85% of app downloads were from the official marketplace. User reviews tend to be frequently used to determine if an app is trustworthy or not. Although it was noted this may cause incorrect privacy decisions as an app can be popular and invasive. Brand name recognition also played a major part in contributing to the user's decision-making process. Price was a major consideration as users tended to have significantly more free applications than paid. Few participants used factors such as user agreements indicating reviews and recommendation were used as the indicators of trust. Overall users were worried about smartphone privacy but did not have the tools to make better privacy decisions.

In a survey conducted by Felt et al. [174], 99 risks were presented to users to determine what smartphone risks were of most concern. Mechanical Turk was used to advertise the survey and users were paid $1.00 to complete it. Over 13 days, 3,115 users responded. It was found that users were really concerned with text messages, e-mails, photos, contacts list and call history being accessed. While location was of concern, it was less of a concern than aforementioned types of data being accessed. Most users indicated that they would remove apps that access sensitive data while a few users indicated they would contact authorities or take legal action. The survey clearly communicated the risks of using certain apps so the participants understood the potential privacy ramifications. In practice, with current platforms it is hard for a user to detect current privacy threats apps pose.

Felt also performed a user study aiming to determine user attention, comprehension and behavior of participants [175]. It was found that current protection systems do not aid users in making correct privacy decisions. An Internet survey of 308 participants and a laboratory study of 25 Android users was conducted. The demographics of participants was representative of Android users. The comprehension of the reading SMS permission was low with only 31.2% responses being correct. The survey was designed to be completed on Android phones and was nine pages long. Permissions dialog boxes were displayed to users where they had to answer what types of functionality they were granting the app.

In the laboratory study conducted in [174], it was concluded that permission warnings do not help a majority of users in making correct security decisions. Non-technical found permissions confusing and hard to understand. Only a minority of users canceled installations of apps due to permission requests. A series of questions were asked and participants were promoted to find and install applications. It was found that only 17% of participants actually looked and paid attention to the permissions. The number of users who were aware of permissions but did not look was 42% and the remaining 42% paid no attention to the permissions.

A comprehensive study was conducted by Mylonas [176] drew the same conclusions as Felt [174] using a different methodology and expanded focus on user attitudes as opposed to user comprehension. The study aimed to determine if users enable security controls, do users consider security while downloading apps and are official app repositories trusted. The demographics of the survey included 458 smartphone users with a minimum age of 15 and a maximum age of 52. The demographics were skewed by age as 81% of respondents were aged between 15 and 30. There was a fairly even distribution of users who considered themselves security savvy at 56.3% versus 43.7% that do not consider themselves security savvy.

In contrary to findings related to the security of app repositories, 76% of users believed apps coming from official repositories were trustworthy. Furthermore 54.6% of people were unaware any testing of apps took place by the maintainers of the repository. The results indicate that the repository simply being associated with a popular brand is enough for users to trust it. It has been found that if a dialog interrupts a task a user is attempting to perform, they are more likely to skip over it. Users did not pay attention to or inspect security messages and warning dialogs. Similar results were found with EULA license agreements presented to users [8].

Survey participants in Mylonas' study disregarded security issues when it comes security and privacy. Smartphone users concerned with privacy spent more reviewing the app. Usefulness was considered the influential app selection criterion with 58.5% of survey participants. Only 3.5% of respondents did not install an app it showed privacy invasive permissions at installation time. Attitudes towards pirated apps that were also studied, they commonly contain malware and are spread through unofficial app repositories that promise users popular apps for free. These unofficial repositories circumvent many of the security features on the platform. 60.7% of respondents reported a preference to pirated applications. Participants in the survey using platforms of wall-gardened models were less likely to use pirated apps than platforms that provided more flexible options to install apps.

Mylonas also reported that the study also suggests that smartphone security controls are poorly adopted by users. Security software was only adopted by 24.5% of survey participants while 85.8% of the same participants used security software on their desktop PCs. Furthermore just 34.4% believed that security software on the mobile is essential. This could be as a result of the platforms applying restrictions making it hard for third party app developers to create software that provides additional security as well as causing the user possible inconvenience with additional restrictions imposed on features such as access to further party apps.

Egelman et al. discovered that there is a market of users that have concerns for app privacy [177]. Concerned participants willing to pay a premium for less sensitive apps. A quarter of study participants indicated they are willing to pay a $1.50 premium for a more privacy aware app. Users are more likely to weigh up privacy between apps when they are able to compare permissions easily across apps.

## 2.8 Regulatory Policies

There has been a number of legislative proposals that will directly impact the implementation of smartphones when they come into effect. Legislative frameworks are necessary to protect individual data privacy effectively. The adoption of technological solutions that protect user privacy is often dependent on the political will of organizations to adopt them. Compliance with regulations can be difficult to implement without technological solutions. Thus, both regulatory compliance frameworks and technological solutions are both required to protect individuals.

Regulatory policies with the aim to protect individual data privacy have been proposed and enacted in various states worldwide. These policies provide added protection to individuals through various compliance practices that need to be adhered to by organizations with penalties issued by the state to those that are non-compliant.

In this section, we discuss the United Nations proposed articles to protect digital user privacy. Global legislative frameworks in privacy protection can vary immensely. The European Union has strong data protection laws and has recently passed new legislation that imposes additional protections related to new technologies.

### 2.8.1 United Nations Articles

An individual's right to privacy is declared in the United Nations charter of human rights in Article 12 [178]. The article states that *"No one shall be subjected to arbitrary interference with his privacy"*. This can be interpreted to expand to digital privacy where inferences should not be made about an individual based on data collected about them.

Recently in December 2013, the United Nations proposed the resolution 68/167 named *"The Right to privacy in a digital age"* which clarifies the view of the UN on the issues surrounding digital privacy. In the resolution, it was reaffirmed that there should be no unlawful or arbitrary interception of communications as well as the collection of data. The UN considers invading digital privacy as a highly intrusive act that limits freedom of expression and contradicts the tenets of a democratic society. There were also concerns noted that such inferences can lead to public security being put at risk.

Smartphones collect almost a complete digital profile of an individual. Access to the data contained in a smartphone violates the right of user privacy. Furthermore, apps and services that collect this data and send it to data warehouses for the purposes of data mining can also potentially infringe on this right if care is not taken to anonymize the user data.

### 2.8.2 European Union Legislation

The European Union has amongst the strongest protections worldwide for the handling of data privacy and has recently expanded its legislation to take into account new technologies such as smartphones and social networks. Initially the Data Directive proposed by the EU in 1995 had to be adopted into law independently at the member states. Recently this directive has been expanded to the General Data Protection Regulation (GDPR) which was adopted in April 2016 and will apply to all member states without having to be passed into law in 2018. This legislation will have implications on how smartphone providers handle data.

An individual that is the data subject now must provide consent before any data can be collected and this consent can be withdrawn at any time. Furthermore, multiple consents are required for different processing activities. For example, if user data is used to train a model, additional consent may be required. This requirement applies even if the service is free. Many services use the freemium model to generate revenue from data collected, this will prove to be more difficult. Smartphone platforms and deployed apps at present do not clearly communicate to the user what data is going to be obtained and for what purposes. When these laws come into effect, platforms will need to take steps to ensure they comply in order to distribute and sell products in the EU.

Information is required to be provided to an individual of all the types of processing data may undergo. This included aggregation or simply being backed up on another form of media. How transparency will be best achieved is still being worked through by the legislators. This may well extend to how data is processed within smartphone apps. It will be required that a user needs to be informed that data may collected and sent to remote data warehouses. In Chapter 7, we demonstrate our proposals to address transparency.

The framework distinguishes between personal and sensitive data. This applies to all data collected that can lead to the identification of an individual either directly or indirectly. Processing sensitive data such as genetic and biometric data is stringent to stricter controls. This can apply to smartphones that collect fingerprint information to unlock the device and make purchases. Further clarification is being proposed on how only data such as the handling of device identifiers and IP addresses. In Chapter 6, we demonstrate how to indirectly identify a user with diagnostic data.

Personal data of a data subject can be processed so it cannot be attributed to a data subject without the use of additional information. This additional information must be kept separately and subject to measures to ensure it cannot be used to attribute the data subject. Pseudonymisation is encouraged by the legislation when data is being used for research purposes. As we demonstrate throughout this thesis, it is possible to infer insights from data considered non-sensitive. When performing psudonymisation, organizations will need to take extra care.

Organizations will need to clearly communicate to its customers if a security breach occurs where data may be stolen. Considering smartphone users do not always provide contact information within the app. Developers may need to implement a mechanism to push notifications to the device to alert them if any unauthorized data access has occurred. It will be expected that organizations build in data privacy by design and not treat it as an additional measure later in the process. For example, all smartphone apps must encrypt any data that is stored on the device irrespective of the data type. There will be implications for developers as additional investment will be required to make apps compliant.

Modern individual data rights include the right to have data erased as well as the right to data portability. An individual can request an organization erase all their data. This includes digital, hard-copy and any data that resides in any backups. Furthermore, an individual can request a copy of data stored at any time and it needs to be provided by the organization in a popular format such as a PDF. Smartphones collect a wealth of information, however this information is stored in a format internal to the device and not easily accessible unless a person has technical skills. Smartphone manufacturers will need to provide some sort of export functionality. In addition, data sent to remote servers will need to be removed upon user request.

## 2.9 Conclusions and Discussion

In this chapter, we presented the state of the art of privacy research that directly impacts modern smartphones. A strong focus was placed on dataset privacy, location/trajectory privacy, privacy protection in current smartphones, user attitudes towards privacy and regulatory frameworks. These privacy research disciplines are often pursued independently and are relevant to smartphones. Thus, we aimed to explore the relevance of these fields and how they apply to researching smartphone privacy.

The foundations of information privacy were explored. Privacy was once implicit as it was difficult to collect information about an individual. The transition from implicit privacy too little privacy begun from the time of when the printing press and photography was invented. Privacy continued to erode with every iteration of information technology with electronic databases allowing for easy storage and retrieval of data, desktop PCs facilitating users creating and capturing their own data and the Internet allowing for content to be shared almost instantaneously. The latest iteration is the smartphone, providing all the functionalities of previous iterations of information technology in addition to more sensitive sensors such as location in the one device. Smartphones like every information technology before it has presented new and unique privacy challenges.

Dataset privacy was considered from the moment databases became commonly used. Researchers considered that databases provided a unique opportunity to solve problems with knowledge that could be derived but struggled to find a way to do it in a way that did not compromise user privacy. The $k$-anonymity model was one of the first cornerstone works that provided a way to share datasets that can be used to run queries without compromising an individual in the set. The premise being to ensure a person is indistinguishable from $k$ people in a set for quassi-identifier attributes. Extensions to these works included $l$-diversity and $t$-closeness. Differential privacy was later proposed and provides a more mathematically rigorous guarantee of privacy. It uses mechanisms to add noise based on probability distributions to ensure individuals in a dataset cannot be identified. Two such mechanisms in common use are the Laplace and exponential mechanisms.

Exchanging datasets in a private manner is evidently difficult even for large multinational companies with famous attacks occurring on datasets over the years. Individuals in the Netflix rating dataset were able to be identified by performing a linkage attack with the IMDB database. AOL search data was not correctly sanitized using best practices and the search history of users were identified. The appeal of crowd-sourcing solutions will likely lead to more breaches in the future.

The spatial nature of mobile devices coupled with positioning functionality led to location privacy to be considered as a key issue. We explored potential attacks as well as protection mechanism. Location privacy was once considered a potential barrier to entry for smartphone adoption by mainstream society. Inference attacks were presented that can determine a user's home location and significant places. Protection mechanism are often based on spatial $k$-anonymity or make use pseudonyms. While theoretical solutions exist, many LBS providers do not implement the necessary functions to use them in practice. Navigation software now available on smartphones is commonly displacing traditional satellite navigation systems. This leads to a user's traveled routes being exposed and promoted the research discipline of trajectory privacy. One such attack we presented was using routing instructions to reconstruct the path a user traveled along. Protection techniques rely on trajectory $k$-anonymity as well as confusion.

Sensitive data stored on smartphones is of potential value to researchers. There are many standard data mining approaches that are in common use such as SVM and neural networks. We demonstrated possible techniques to mine for data in a secure manner without the user actually exposing their data. This allows for the benefit of data mining to be achieved without the drawback of violating a user's privacy. Many of these methods involve secure computation techniques proposed by the cryptography community. It is likely that these methods will be implemented on mobile platforms in the future.

The design of smartphones themselves directly impact on privacy protections. Early smartphone operating systems were really restrictive while modern platforms relaxed the requirements to encourage third party development. The most popular platforms at present being Android and iOS take two varied approaches. Android relies on a system of permissions that apps require from the user to access functionality while Apple manually review the apps. Each approach has their own drawbacks. In the case of Android, many users do not comprehend the permissions or simply agree apprehensively for the convenience of what the app provides. The manual review process used by Apple for iOS requires users to have faith that the process is adequate without having exact knowledge of what is performed.

Regulatory protections were considered as these are often needed to motivate developers to safeguard user privacy. New laws coming into effect will directly impact the design of smartphones. The current generation of devices are not compliant and will need to adapt if the manufactures want to continue to sell within certain markets, especially in the EU. New protections offered include full disclosure of what data is being captured and stored by an organization and the right to request a copy of the data in an easily comprehensible format as well as the right to erase the data from organizational stores via requests.

Many studies into user attitudes of privacy show that users are concerned but are willing to sacrifice their privacy for convenience or financial compensation. There is evidence to suggest that users understanding the potential implications of giving away privacy such as being able to be physically located reduces the likelihood a user will share this information. Other studies have shown there is a segment of users willing to pay a premium for a more secure phone leading to more motivation for research in this space to meet the said privacy requirements.

A gap in the literature is the potential impact of the nature of data being explored. Data that appears to be innocent upon first glance can reveal sensitive information a user may not wish to share once it has undergone thorough analysis. In this thesis, we aim to bridge that gap by presenting algorithms and techniques that reveal sensitive insights about a user via data captured on a smartphone. Given this knowledge, we also explore ways to protect user smartphone data at an operating system level. This understanding is needed to ensure the next generation of smartphones are more privacy conscience.

# Chapter 3

# Trajectory Inference Attacks on Smartphone Users

In Chapter 2, techniques to protect location and trajectory privacy were presented. In this chapter, we explore how direct location data can be inferred indirectly using query results data. Query results appear to be anonymous by not revealing the exact user location, however the location of the query results act as a proxy for indirect inference. Most existing studies focus on protecting trajectory data through obfuscating, anonymizing or perturbing the data with the aim to maximize user privacy. Although such approaches appear plausible, our work suggests that precise trajectory information can be inferred even from other sources of data.

Smartphones can position users with accuracies of up to 5m. Third party apps can retrieve location data and build a dataset of an individual's location tracks over time. Trajectory data does not only show the location of users over a period of time, but also reveals a high level of detail regarding their lifestyle, preferences and habits. The potential exchanging of sensitive location data is of concern. Datasets are often exchanged to improve the performance of search algorithms or derive marketing insights to better understand what contextual ads to show a user.

Trajectory privacy has become a key research topic when sharing/exchanging trajectory datasets. The query results in these datasets are commonly exchanged under the assumption they are anonymous if location tracks are removed. We consider the case in which a location service provider (LSP) only shares POI query results of users with third parties instead of

---

**This chapter is based on**
**Tell Me What You Want and I Will Tell Others Where You Have Been - CIKM 2013**

exchanging users' raw trajectory data to preserve privacy. We found that the query results provide enough location information to still locate where a user is at a certain time and the route they travel along.

We develop an inference algorithm and show that it can effectively approximate original trajectories using solely the POI query results. The algorithm demonstrates high performance in reconstructing the route a user travels along and their location at a given time. The advent of services making use of continuous queries makes this inference technique feasible on real world datasets. This finding prompts data providers to take additional care when exchanging query results.

## 3.1 Introduction

Many applications, e.g., traffic management, urban management and geomarketing, gain substantial benefit through mining such sources of trajectory data. However, sharing this data in a raw format with third parties may incur serious privacy threats. Particularly, if inappropriately protected, such data may turn into powerful means of privacy invasions [179, 180, 89] such as location-based spams, physical threats, or inference attacks [74]. Findings of a recent study indicates that a large number of location based service (LBS) users are concerned about their privacy, which places a great impediment to sharing trajectory data and the growth of LBSs in general.

Various approaches in the literature focus on preserving privacy prior to publishing/sharing trajectory data [104, 181, 182, 142, 183]. The authors in [144, 184] propose publishing a *k-anonymous* trajectory dataset to make a user indistinguishable from $k-1$ other users [33]. Other studies adopt cloaking/obfuscation techniques to coarsen the spatial and/or temporal features of a trajectory [115] before publishing it. The authors in [107] propose an approach that adds dummy trajectories to keep the data private.

To motivate our work we consider the case of an LBS provider who sanitizes a dataset by omitting all sensitive attributes from the dataset [13]. In other words, instead of anonymizing or obfuscating trajectory data, the LBS provider removes all GPS tracks of its users along with their identity, presuming this will provide maximum privacy. This means that only the *results* of user issued queries, which is generated and *owned* by the LSP, remain in the database when sharing it. Recent features built into modern smartphone platforms location

APIs also make it possible for third-party apps to access continuous location updates even when the app is running in the background [185, 186]. Thus, it is possible for both service providers and app developers to have access to sensitive location data.

To identify the risks of such a plausibly bullet-proof practice, we consider a scenario where users continuously request the closest POIs to their position. For instance, a user issues a query like *"Where is the nearest gas station to my path?"* or *"Send me the closest Italian restaurant?"*. POI queries do not necessarily need to be issued by a user manually. LBSs commonly make use of continuous queries that constantly require positioning updates. Location recommendation systems [187] and the "nearby friends" feature commonly available in social networks makes use of continuous queries. If detailed GPS tracks are removed from the database, the LBS provider may lead others to believe that it is safe to exchange the POI query results – a set of POIs ordered based on querying time.

In order to demonstrate the vulnerability of sharing LSP query results with a third party, we develop an algorithm to perform an indirect inference attack on query results. The algorithm infers individuals' trajectories using the response of LSP to the requested POIs. The trajectory reconstruction is performed without using GPS tracks captured by the LSP but instead, the requested POIs sequence. As background knowledge we assume the availability of road types including how central a road is (e.g. main road), how often a road is used by motorists, the speed limit and basic transportation information about the area.

To the best of our knowledge, our work is the first to attack trajectory privacy without directly using trajectory information (either fine-grained or coarse-grained). The accuracy of our approach suggests that indirectly inferred paths are sufficiently precise to raise serious privacy concerns. By demonstrating what indirect trajectory inference attacks can achieve, we show that there is an urgent need to address the privacy implications of LBSs when exchanging trajectory datasets even when the provider omits the sensitive location data.

## 3.2 Background

Generally, analyzing data in order to gain knowledge about a subject in an adversarial manner is known as an *"inference attack"* [87]. A key work to highlight the potential of location data in predicting users' movement behavior was given in [90]. Ashbrook et al. have used GPS data of mobile users to determine significant places being visited by them [90]. To underline the risks of leaked location data [87] used real GPS data of 172 subjects and found each person's home location with a median error of about 60 meters. Finally, [87] identified people based on their pseudonymous location tracks using simple algorithms and a free Web service. In contrast, our algorithm can estimate a user's trajectory without using the GPS tracks stored by the LSP.

In another study, [99] succeeded at reconstructing an unknown trajectory using its distance to a few fixed trajectories. In order to achieve this, the authors have introduced speed limit and known trajectories as a background knowledge used by an adversary. Similarly, assuming it is possible to observe a user's movement behavior in public places, or even inferring a couple of her visited spots using social networks, weblogs etc. [188] used some snapshots of a user's trajectory as adversary's background knowledge. [188] proves that this *"general world knowledge"* can breach user's privacy with a high probability regardless of how much attempt is being taken to anonymize or cloak her location data. These two works clearly show the potential risks stemming from combining the available background knowledge along with the mutual distances released for analytical purposes.

Probabilistic map-matching methods such as [189] attempt to construct a trajectory from a set of discrete location samples. Map-matching techniques heavily make use of shortest-path algorithms to find the most likely path between samples. These techniques work well provided the granularity of samples is fine enough to reduce possible paths a user could have traveled along. Current map-matching algorithms also suffer from high time complexities making it difficult to apply at scale. Our problem domain assumes sparse location samples so these techniques are not directly applicable.

Voronoi diagrams [190] are widely used in computational geometry. The region of a Voronoi cell is comprised of all points that are closet to the Voronoi point corresponding to the respective cell. Its structure lends itself nicely to answering nearest neighbor queries as each POI within a Voronoi cell will correspond to the closest POIs. The structure has also found practical applications in a variety of other fields.

## 3.3 Problem Definition

### 3.3.1 Closest POIs Database

POI queries are a common applications of LBSs. An LBS user sends her current location accompanied with a query asking for her closest points of interest, e.g., the closest gas station, Italian restaurant, etc., and the LSP returns a set of points as query result. The query database records are in the form of ($ID$, $\mathcal{P}_{\mathcal{T}}$), where $ID$ determines a user. $\mathcal{P}_{\mathcal{T}}$ is the result of successive POI queries along the user's trip. In other words, $\mathcal{P}_{\mathcal{T}} = \{p_1, p_2, ..., p_n\}$ where each $p_i$ represents a the closest point of interest for the $i_{th}$ query. Note that $ID$ is not necessarily a user's actual identity, but rather a unique identifier such as a pseudonym.

### 3.3.2 Adversary Model

An LSP may remove both user identity and location information as the sensitive attributes from the query database, presuming this would guarantee the user's privacy. This supposedly anonymized database is then shared with third parties. We suppose that an adversary is any third party with whom this query database is shared. One example of this occurring recently is researchers were able to mine check-in locations from Foursquare users to construct a dataset and made it publicly available [191]. In addition to the closest POIs database, we assume that the adversary has one of the following two types of background knowledge about a transportation network:

**Edge Centrality:** The adversary may consider each street as an edge in a graph and assume that the more central an edge is, i.e., how frequent an edge occurs in a set of candidate paths, the more likely it is that a user travels along it. Hence, edge centrality can be modelled introducing a weighting function. Generally, main roads are more likely to receive higher weights.

**Edge Frequency:** The adversary may have access to the trips users have taken in the past. In this model, we assume that the adversary assigns a higher weight to more frequently travelled edges. The count for a specific edge is incremented for every trip that it could be part of to account for the GPS error. In addition, we assume that an attacker also has the following information:


(a) GPS Error and Line Interpolation (b) Accuracy $\approx 0.91$

Figure 3.1: Proximity Circles to Measure Location Privacy

**Maximum Velocity Bound:** The adversary may also assume that there is a maximum velocity with which a user can travel between two subsequent time stamps. The velocity of a user at a given time can be estimated based on the maximum speed limit of a road.

### 3.3.3 Attack Success

GPS points do not uniquely identify the roads a user has taken, so the actual trip is generally not known. Thus, conventional trajectory similarity measures such as the Hausdorff distance and DTW are not suitable to assess the success of our algorithm. Take Figure 3.1a as an example, where the GPS logs do not overlap with the actual path due to the measurement error. Using linear interpolation between GPS points (the dashed line connecting points) does not provide a robust means of inferring the original path, and as can be seen in Figure 3.1a, this interpolation may barely overlap with the underlying road network. Even map matching cannot resolve such a situation because a GPS point cannot be uniquely mapped to a single edge.

To address this issue, we determine the closeness of an inferred path to the original one through proximity circles. We draw circles with $r$ radius around the location of each GPS point (Figure 3.1a) and estimate the attack success based on the circles. More specifically, the overlapping percentage of the inferred path within $r$ meters to the original path determines the success of the inference attack. This is measured by the number of proximity circles that are *visited* by the inferred path segments divided by the total number of circles.

Figure 3.1b shows the example of an inferred path as well as the proximity circles representing the original GPS logs. The inference accuracy is measured using the above equation and is $0.91$. This metric is useful in evaluating how well the algorithm works in identifying the band a user is travelling along.

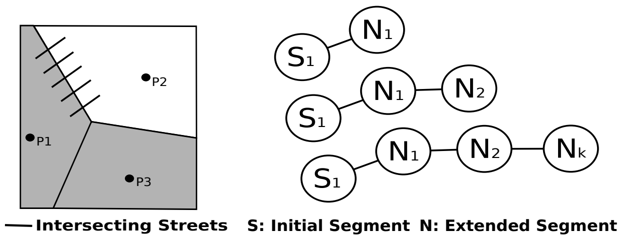



Figure 3.2: Generation and Extension of Initial Path Segments

## 3.4 Indirect Trajectory Inference Algorithm

Our proposed inference algorithm utilizes the location of query results to generate a Voronoi diagram, which for a given set of points, $p_i \in \mathcal{P}$, divides the space into a number of cells (regions) such that all the points in any cell, $c_i$, are closer to the corresponding $p_i$ than to any other $p$. Voronoi diagrams are widely applied in nearest POIs problems. For the set $\mathcal{P}_T$, we generate a Voronoi diagram $V$, and retrieve a set of *candidate paths* that travel from one Voronoi region to the next. However, since Voronoi cells can be quite large in many areas, the number of these candidate paths is not restrictive enough to reconstruct a unique trajectory and needs to be further reduced. Therefore, maximum velocity bound and edge centrality have been employed in this work to infer the most likely path that was taken by the users.

### 3.4.1 Indirect Path Generation

To generate our initial paths, we propose an incremental search algorithm that iterates through each pair of points. On the first iteration, the Voronoi edge between the first and second point is determined and every path segment that intersects with the Voronoi edge is retrieved. This is illustrated on the left of Figure 3.2. All streets that intersect the Voronoi edge between $p1$ and $p2$ are retrieved. The initial retrieved path segments are considered as the starting path segments for the candidate path(s). The initial segments are then passed to a function which retrieves all the connected path segments to the initial one and add them to the respective candidate path (Figure 3.2). Once a set of paths is generated, the last path segment of each path is checked to ensure that it is in the destination Voronoi cell, otherwise the entire candidate path is discarded.

In addition, assuming a maximum velocity bound, $max_v$, we can compute the maximum distance, $d$, that a user could have traveled between two consecutive timestamps. Therefore, the length of the path is checked and only if its difference with $d$ is less than a threshold, $\delta$, it is added to the set of candidate paths. Since the beginning of a path segment is not necessarily the start point of the trajectory, we assume the generated path may be slightly longer than $d$, i.e., $\delta$ longer.

### 3.4.2 Candidate Path Selection

To select a single path from the set of generated candidate paths we derived a weighting function to rank the paths based on edge centrality, and then select the top ranked candidate path as the path representing the user's trajectory.

The weighting function counts the frequency of path segment occurrence in each candidate path and stores this value in a hashtable. The weight for each candidate path is calculated by the summation of each path segment length divided by the length of the whole candidate path which is then multiplied by the frequency value of each path segment stored in the hashtable divided by the total number of candidate paths.

$$w_{path} = \sum_{i=0}^{n} \frac{pathSegmentFreq_i}{n} \times \frac{pathSegmentLength_i}{totalLength_i} \tag{3.1}$$

The weighting function returns the path that contains the greatest overlap with other paths in the candidate path set. Therefore, edges that are more commonly used in a set of candidate paths are favored over edges that are not. Provided users take fairly direct routes to their destination, this weighting function works well.

## 3.5 Experiments

### 3.5.1 Dataset

In our work we employ the GeoLife trajectory dataset[1] to evaluate the performance, i.e., accuracy, of our inference approach. The GeoLife dataset consists of more than 17,000 trajectories that have been collected by 182 individuals over three years. We focused on a smaller part of the city of Beijing and retrieved those trajectories that fully reside inside this part. In total we ran our inference algorithm on 279 routes.

[1] www.research.microsoft.com/en-us/projects/geolife

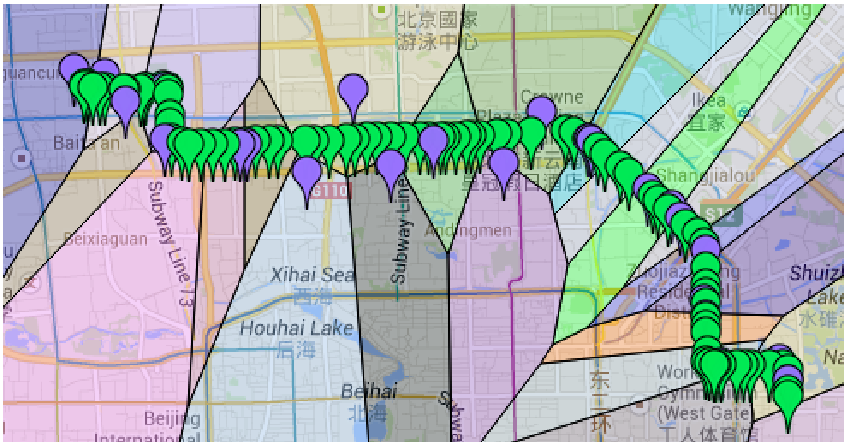



Figure 3.3: Voronoi Diagram Generated for a Path

### 3.5.2 Implementation

We generate random POIs in the city of Beijing and stored them in a database as our $\mathcal{P}_T$. These POIs are generated uniformly inside the boundary of Beijing and they are then mapped to their closest road segment. In order to evaluate the performance of our inference algorithm for different scenarios, we create four POI batches of 400, 800, 1600 and 3200 points to reflect sparse and dense areas. For example, 1600 POIs equate to 4 POIs per square kilometer.

In the road network, there are many path combinations a user can travel along to get to a destination. This leads to a large search space and an inefficient search process. To reduce the search complexity, we employ a pruning method that discards any combinations that terminates at the same road while expanding paths between Voronoi cells. Moreover, due to the geometric nature of Voronoi diagrams, individual Voronoi cells can become quite small and may create a case where some paths overshoot the cell and lead to zero candidate paths. In order to compensate for this case, the algorithm allows paths to continue to expand even if they did not terminate in the current Voronoi cell. However a path is finally removed if it does not end in the next Voronoi cell in the next iteration. In our implementation we use the OpenStreetMap[2] data to generate the road network graph. A web interface is also constructed to visually view the data using PHP, Javascript and the Google Maps API. Javascript is used to implement the trajectory inference algorithm and an OpenStack cloud

[2] www.openstreetmap.org

| POI | $r = 50\text{m}\ (\%)$ | $r = 100\text{m}\ (\%)$ | $r = 250\text{m}\ (\%)$ | $r = 500\text{m}\ (\%)$ |
|---|---|---|---|---|
| 400 | 27.73 | 39.10 | 51.83 | 64.74 |
| 800 | 35.10 | 47.97 | 61.31 | 73.76 |
| 1600 | 39.00 | 53.90 | 69.63 | 80.84 |
| 3200 | 36.32 | 49.74 | 64.38 | 75.37 |

Table 3.1: Trajectory Inference Experimental Results using Edge Centrality

| POI | $r = 50\text{m}\ (\%)$ | $r = 100\text{m}\ (\%)$ | $r = 250\text{m}\ (\%)$ | $r = 500\text{m}\ (\%)$ |
|---|---|---|---|---|
| 400 | 32.15 | 44.52 | 58.09 | 71.31 |
| 800 | 38.03 | 51.93 | 65.85 | 77.79 |
| 1600 | 41.07 | 56.44 | 71.62 | 81.40 |
| 3200 | 37.97 | 52.45 | 67.70 | 77.97 |

Table 3.2: Trajectory Inference Experimental Results using Edge Frequency

computing environment is utilized to run the inference algorithm. Moreover, the Bower-Watson algorithm is employed to compute the Voronoi diagrams. An example GeoLife path is also illustrated in Figure 3.3, where the purple (darker) flags illustrate POIs along the path and their respective Voronoi cells. The green (lighter) flags show the original GPS logs.

### 3.5.3 Experimental Results

We measure the accuracy of our inference algorithm as the number of proximity circles that are visited by the inferred path divided by the total number of circles (Section 3.3.3). To estimate the inference accuracy of our approach, we consider varying radii, $r$, in meter to generate proximity circles, where $r \in \{10, 50, 100, 250, 500\}$. Table 3.2 shows the performance of our inference algorithm for varying POI densities. Although the overall accuracy is low for very small radius of 10 meters, our results show that for more realistic buffers and higher densities our approach is successful in accurately estimating a user trajectory and can achieve an average accuracy level beyond 80%. When POI densities are excessively high, this can cause accuracy to drop slightly in our approach as the Voronoi cells become smaller causing additional but possibly legitimate routes to become pruned. However, overall the trend is that the increase in POI density and for urban areas with higher POI densities such as city centers, the inferred paths get closer to the original paths incurring higher privacy risks for a user.

In order to understand if access to the trip patterns of users can increase an attacker's ability to infer a user's original path, we ran experiments using edge frequency instead of edge centrality. The edge frequency (see Section 3.3.2) is computed on the basis of actually travelled trips. Our experiments show small gains in terms of accuracy but also demonstrate that this additional knowledge does not significantly improve an attacker's ability to infer a user's path. Our findings show that in either case the ability of an attacker to infer a user's path based on POI information is high.

## 3.6 Conclusions and Future Work

In this chapter, we presented an algorithm that only employs a set of POIs to indirectly infer a user's trajectory. Our results suggest that even coarse location information allows us to approximate a user's trajectory within dense urban areas with high accuracy. Therefore, exchanging query results of LBS users instead of their tracks does not offer adequate privacy protection.

While our inference attacks are effective in areas with high POI densities, there are still a number of directions that are likely to make the overall inference strategy more effective. We assume in our work that for every request there is only the closest POI available, however, an LSP usually provides users with several POIs for a single request. This information could be encoded as a higher-order Voronoi diagram that leads to smaller cells and thus should enable a more refined attack strategy.

In our work we have used positive information, i.e., information directly shared by a location service provider. However, another location service provider (or adversary) could also have the information about all the POIs that were not revealed because they were not among the closest POIs. Since the overall number of POIs is much larger than the number of POIs returned as a query result, the underlying Voronoi diagram may result in smaller cells, which in turn should improve the accuracy of an inference attack algorithm.

# Chapter 4

# Locating Smartphone Users Indoors

In Chapter 3, we presented how to reveal user location indirectly from query results. In this chapter, we demonstrate how to reveal user location indoors via Bluetooth using a localization scheme we developed. Smartphone users may not want to reveal their location in an indoor environment. For example, an employee of an organization may be happy to make knowledge of where they work public but do not want their exact location within an office complex to be known.

Many users also leave on Bluetooth to connect to headsets and other smartphone devices. Our localization scheme indirectly reveals a user's location by comparing the signal strengths of audible beacons and other devices to pinpoint the user. This chapter demonstrates how a numeric value indicating signal strength can be used to narrow down a user's location.

Signal strength data is considered to be not sensitive and can be captured by being within range of another device. Smartphones are equipped with Bluetooth which uses radio signals to communicate. Nearby devices can perceive these signals and determine the relative signal strength. Also revealed with the signal strength is the device name. Many smartphone users set this device name to their name. In this chapter, we present a localization scheme that can accurately position users.

Localization schemes used for positioning in wireless networks are currently based on either range-based or range-free principles. Both methods when used in isolation have disadvantages that can reduce the accuracy of positioning estimates. We propose a unified approach that combines the strengths of both methods while overcoming their limitations.

---

**This chapter is based on**
**Combining Range-Based and Range-Free Methods: A Unified Approach for Localization - SIGSPATIAL 2015**

Range-based methods rely on taking reliable measurements in which geometric techniques are then applied to locate a node. These techniques are very susceptible to imprecision in captured measurements. In contrast, range-free methods do not consider the actual numerical value of a positioning device, rather reading magnitudes compared to other measurements provides the most reliable information. These comparisons are not always reliable and may lead to an accumulation of error, particularly in large spaces.

Reliable schemes rely on beacons of known locations, geometrically it can be seen that a greater number of beacons can lead to better granularity in positioning estimates. Our unified approach mitigates these effects by first using a range-based method to determine an approximate location which can be then used to reduce the search space. A range-free method is then employed to refine the positioning estimate further within the reduced space.

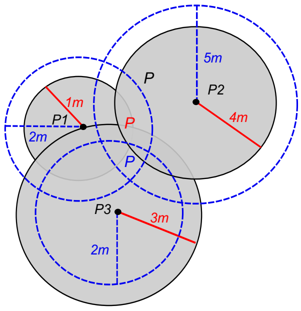
Deploying a sufficient number of beacons required to achieve high precision is not always practical, thus we introduce the concept of virtual beacons that estimate signal magnitudes at points to improve granularity. Our experiments show the mean estimation error improves when applying our localization scheme to a Bluetooth system and improves further with virtual beacons.

## 4.1 Introduction

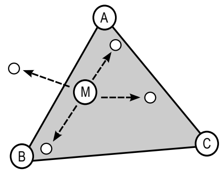
Positioning mobile devices in a wireless sensor network requires advanced localization schemes. Current algorithmic principles are either range-based or range-free. Neither method in isolation currently provides a general solution that works well in a wide range of scenarios, especially for indoor settings.

Localization schemes require anchor points or location beacons of known locations in order to perform location estimation. Common estimation techniques make use of trilateration, triangulation, fingerprinting and proximity principles [192]. Trilateration measures the distance from multiple reference points while triangulation derives the angles. Fingerprinting requires measurements to be manually taken and stored offline. A sensor will compare its readings to that of the measurements taken earlier. Proximity commonly used in binary sensors relies upon a dense grid of antennas and a location is estimated based on which sensors are nearby.

Range-based methods are considered to be capable of achieving fine-grained precision.




(a) Trilateration




(b) APIT InToOut Case

Figure 4.1: Potential Issues Range-Based and Range-Free Methods Encounter

Measurements are first taken, then signal propagation models or timing methods are employed to estimate distances. Many assumptions are made about the environment in which they operate. In signal propagation models, constants need to be adjusted depending on the setting the positioning device is in. In timing methods, distances between location beacons need to be known ahead of time. Thus, while providing good positioning accuracy in known environments, in dynamic environments these models do not always hold. In contrast, range-free techniques do not make any assumptions that such information is available or even valid. Range-free techniques use binary decisions, e.g. is a user in range or not or is a user within a triangle. While range-free methods are robust, they are considered coarse-grained as the level of granularity they provide is bounded by the geometry formed by location beacons.

This chapter introduces a unified approach that takes features from both range-based and range-free principles. By unifying these approaches, the accuracy of the positioning estimate being negatively affected by imprecise measurements associated with range-based methods and accumulation of error across large spaces that occurs in range-free methods can both be mitigated. We use a range-based method to obtain an approximate location, this location is within the three beacons that are considered closest. The location is then used to reduce the search space, for which a range-free method is then applied. Since range-free methods make use of information from other devices, it is suitable for further refinement. This does not necessarily work as well in reverse because range-free methods are susceptible to accumulation errors in large spaces.

We illustrate our reasoning with the following examples. Consider the scenario presented in Figure 4.1a where three location beacons and a user node are present. The actual

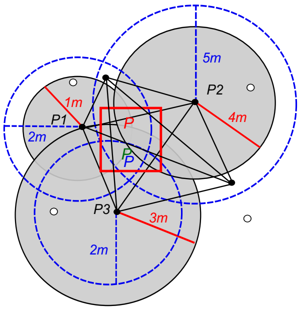



Figure 4.2: Combining Range-Based and Range-Free Methods

distance between the user node and beacons is $2m$, $5m$, $2m$ respectively, however using a signal propagation model the distance calculated is $1m$, $4m$, $3m$. What is known with more confidence is based off the signal strengths, these beacons are most likely to be the three closest. Thus, a node is somewhere between these beacons, however due to high sensitivity of the models we do not assume the calculated position within the triangular area of the beacons is accurate.

Range-free methods make use of information from neighboring nodes to estimate a location. A popular range-free scheme referred to as Approximate Point In Triangle Test (APIT) [193] overlays a geometry of triangles by joining location beacons. The key idea is to be inside or outside a triangle. The test is performed through a comparison of signal strengths of the received signals. More triangles leads to an increase in potential expected accuracy as the position can be approximated through the intersection of all triangles a device is located in. As the number of beacons is typically small, the expected accuracy will be low as well. Through a process referred to as APIT aggregation, the position is estimated to be at the center of gravity of the triangles all surrounding neighboring nodes consider the device to be in. This method is not without limitations, consider the scenario shown in Figure 4.1b. By performing a APIT test, the scheme estimates the node is outside when the node is really inside the triangle. This occurs in cases where a node is close to the edges of triangles. If there are many neighboring nodes clustered in one area, this will also cause a bias. In larger spaces, range-free methods tend to favor where nodes are more central as error accumulates, thus range-free methods work better in smaller regions. Having obtained an approximate location from the range-based method, we only need to consider a smaller region. Provided there are neighboring nodes nearby, this information is used for further refinement.

In our unified approach, a bounding box is created around the approximate location obtained by the range-based method. Refinement is then achieved by using APIT which makes use of information of neighboring nodes. A signal magnitude is compared in relation to the magnitudes of all neighboring sensors to determine if its value is greater than or less than its neighbors. The numerical value itself is not important, the binary result from the comparison operation is the focus when refining the position. In the case where sufficient neighboring nodes are present, this can lead to a significant reduction of error. An increase in node density leads to more values for a sensor to compare. For example, in the case of APIT, the algorithm can use neighboring nodes to determine if a node exists that is closer to a location beacon than the node we are trying to locate. Consider the case presented in Figure 4.2 which illustrates a hybrid of the above two methods. Trilateration results in a position being above the actual position, a bounding box surrounds this point. Triangles are generated using location beacons, APIT is restricted to the search space within bounding box. The position estimate is now closer to the actual position due to the neighboring nodes indicating which of the three polygons the user is closer to.

In coarse-grained schemes such as APIT, additional location beacons can lead to added combinations of triangles which can improve the granularity of positioning estimates. Because actual location beacons may not be available, we propose the use of *virtual beacons* that use real beacons to estimate a possible signal magnitude if a beacon actually existed in a particular location.

Virtual beacons compute their location using the received signal strength of actual beacons on the basis of a path loss model. Provided the virtual beacon locations are close to actual beacons, they are robust to signal strength variations. The range-free component of our algorithm treats actual and virtual beacons in the same way for location estimation. The range-based component only uses real beacons as the measurement phase is much more susceptible to signal noise. A similar principle was proposed in RFID based systems using virtual references to yield improvement [194].

We focus on applying our localization scheme in an indoor scenario because outdoor positioning is commonly achieved via GPS. It was also our aim to use off the shelf technology, i.e. Bluetooth. Devices with Bluetooth capabilities are widely available ranging from headsets and speakers to phones and tablets. Furthermore Bluetooth is available on both modern and older generation mobile devices. Any programmable Bluetooth device can act as a sensor that reads values from multiple beacons. The ubiquity of Bluetooth makes it a good candidate to create a practical, widely available and cost effective indoor positioning systems.

The use of Bluetooth beacons are becoming increasingly popular in the application of indoor positioning as they can interface with modern smartphone apps, relatively easy to deploy and are cost effective for large scale deployments. The Eldheimar Museum in Iceland recently created the automatic museum guide available for visitors to install on their smartphones [195]. The museum guide service provides live navigation instructions and presents content based on the user proximity to key exhibits. Similarly, American Airlines released an app that provides important information based on a customer's location [196]. Beacons are deployed throughout the airport pushing information content to users such as the closest security checkpoint, boarding time and distance to the gate.

To evaluate our approach, we developed a Bluetooth simulator that visually allows the placement of users, beacons and nodes in a room. Our results indicate we can achieve higher positioning accuracy when using our unified approach as opposed to when using either a range-based method and range-free method in isolation. Adding a sufficient, number of virtual beacons also improves estimation accuracy.
Our main contributions are as follows:

- Propose a unified approach to localization making use of both range-based and range-free methods;
- Provide an algorithm demonstrating the effectiveness of a unified approach;
- Introduce virtual beacons to demonstrate how they can be used to improve results;
- Demonstrate that our technique can achieve accuracies of under $1m$.

## 4.2 Background

Localization is considered an important problem in which many techniques have been proposed that can be classified as either range-based or range-free. Recent research has focused on applying schemes in an indoor setting [197, 198]. Early work focused on applying range-based techniques commonly used for outdoor environments in indoor settings. These solutions often require expensive hardware and complex deployments. Later research aimed to find a solution using range-free techniques as they are cheaper and more accessible.

### 4.2.1 Range-Based Methods

Range-based methods consist of two phases, the ranging phase and the estimation phase. In the ranging phase, measurements are taken to estimate distance by communicating with sensors of known locations. The estimation phase uses the measurements taken and applies geometric techniques to estimate a position.

The Received Signal Strength Indicator (RSSI) of a sensor can be used to measure distance. Timing methods including Time of Arrival (ToA) and Time Difference on Arrival (TDoA) can be alternatively used to measure distance. These methods are usually more accurate, however have a greater communications overhead. Sensor clocks need to be precisely synchronized for timing method to work with minimal error.

In ToA, the one-way propagation time is measured between a location beacon and a node. Given the speed the signal travels is known, the distance can be then calculated by using intersecting circles, triangles or spheres. TDoA is a more complex variant, examining the time difference of when the signal arrives at multiple receivers as opposed to using just the absolute time of arrival.

In TDoA, stations are at known locations and broadcast their signals at known times. The difference in distance is measured resulting in an infinite number of locations that can satisfy the measurement. When plotted, these locations form a hyperbolic curve. Taking measurements from a second station combination will produce a second curve which intersects the first. The two curves are compared. A small number of possible locations are revealed producing a fix.

Angle of Arrival methods (AoA) measure the angle between a node and location beacons. Directional antennas or an array of antennas are required. No time synchronization is needed, however additional hardware is required and AoA solutions are expensive to deploy. The angle the positioning device makes between two location beacons is measured, by using this angle to form two lines, the position is the intersection of these lines.

The method used in the estimation phase depends on what ranging method is initially used. When using distance measurements via RSSI or ToA, trilateration can be employed by using circles, triangles or spheres to locate a device. In contrast, triangulation can be used when angles are measured from known points using AoA. The curves produced in TDoA are compared via multilateration techniques.

Satellite-based systems including GPS make use of ToA methods and have proven to be highly accurate for outdoor scenarios. Readings are taken by a receiver from at least four satellites, four is required to perform trilateration in three-dimensional space. Positioning satellites carry atomic clocks which are periodically synchronized with ground stations. While providing the widest possible coverage, satellite-based systems do not usually perform well when applied to indoor scenarios [199] due to the line-of-sight requirement.

The measurements taken in the estimation phase can greatly affect the accuracy of the approach. Small errors in measurements such as distance and time leads to the estimation accuracy dropping off rapidly. While range-based methods are conventionally considered to be a fine-grained method to localization, in the case of our unified approach that is intended for indoor scenarios, we use it to gain an approximate of where a user is located.

### 4.2.2 Range-Free Methods

Range-free methods make little to no assumptions about the environment in which they operate and rely upon error reducing methods such as aggregation. Well known range-free schemes proposed include Centroid [200], DV-HOP [201] and Amorphous [202] and APIT[193] which will be described in detail in Section 4.3.2.

Centroid works under the assumption coarse grained localization information is available from location beacons. The centroid of all the received location beacons in an area is then taken and used as the location estimate [200]. This approach has the advantage of being easy to implement. The following formula is used to estimate the user location based of locations received of all audible beacons ($N$).

$$(X_{est}, Y_{est}) = \frac{(X_1 + ... + X_N, Y_1 + ... + Y_N)}{N}$$

In Figure 4.3, it can be seen that position $P$ has been determined to be the center of the three location beacons considered closest. The authors found that this method works well for outdoor environments with the mean estimation error never exceeded $2m$. In indoor settings, the performance accuracies range greatly between $4.6m$ to $22.3m$.

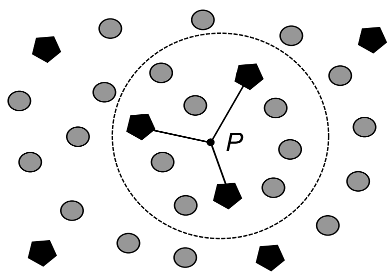



Figure 4.3: Centroid Localization

The DV-HOP and Amorphous algorithms share similarities borrowing from principles in classical distance vector routing. Initially location beacons are searched for by broadcasting throughout the network outwards, initializing a hop-count parameter to 1. Nearby nodes that receive broadcasts track the hop-count in a table, increment it and forward it to surrounding nodes. Initialization of hop-count can be seen in Figure 4.4.

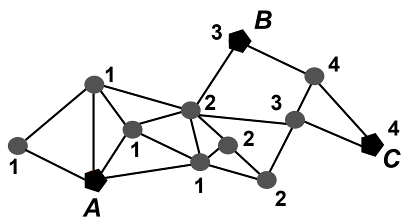



Figure 4.4: DV-HOP/Amorphous Localization

Hop-count is then translated to physical distance, DV-Hop uses the average single hop distance using formula (4.1) . The location of $i$ and $j$ is represented by $(X_i, Y_i)$ and $(X_j, Y_j)$ respectively, furthermore the distance in hops is represented by $H_j$. The result of this formula is propagated to nearby nodes.

$$HopSize_i = \frac{\sum \sqrt{(X_i - Y_j)^2 + (X_i - Y_j)^2}}{\sqrt{H_j}} \tag{4.1}$$

Amorphous converts to physical distance using a more complex method, the calculation used is referred to as the Kleinrock and Slivester formula (4.2) [203].

$$HopSize = r(1 + e^{-n_{local}} - \int_{-1}^{1} e^{-\frac{n_{local}}{\pi}(arccost - t\sqrt{1-t^2})} dt) \tag{4.2}$$

An assumption is made that the density of the network is known ($n_{local}$) so that the $HopSize$ can be calculated offline. The final step of both DV-Hop and Amorphous is to determine the closest three location beacons. by calculating the distance measurements, trilateration can then be performed to estimate a node's location.

Range-free methods are considered to be a robust coarse-grained approach, however in our unified-approach we use range-free methods to refine a position further. Once a localization scheme has an approximate location, range-free methods can use information provided by neighboring nodes to refine estimate further.

### 4.2.3 Indoor Positioning Systems

Indoor Positioning Systems (IPS) aim to provide positioning and tracking of mobile devices or persons. IPS have many practical applications, currently they are commonly used where locating users in a building is required and inventory tracking of expensive equipment. In this section we focus mainly on Radio Frequency (RF) systems as our work is based on Bluetooth which uses radio waves. A solution that is accurate, robust, scalable and cost effective is required for mainstream adoption. Table 4.1 compares different localization schemes across these criteria. In our work we have applied both range-based and range-free techniques to a Bluetooth system, utilizing methods from both ideas to refine location further.

Many IPSs have also been proposed using varied indoor localization schemes. The choice of frequency used is a major determining factor of what scheme a system should employ. In this work, we focus specifically on Bluetooth systems that use radio frequencies (RF). Indoor positioning has proven to be difficult to achieve. In free space, radio multipath models work well in determining distance. This does not usually generalize to indoor situations except for in controlled settings [204]. Indoor environments pose difficulties due to severe multipath propagation and signal attenuation caused by various objects. In Figure 4.5, the signal is split into two paths because it reflected of an object, it is then weakened as it goes through the wall.

Early work focused on centralized approaches that built upon client-server architecture. A system called BIPS [205] used computer workstations as beacons of known locations. The beacons constantly scan for mobile devices and readings are sent to a server for processing. Similarly Bluepass [206] applied the same approach using USB Bluetooth adapters connected via USB cable as fixed stations as opposed to computer workstations.

| **Technique** | **Accuracy** | **Robustness** | **Scalability** | **Cost** |
|---|---|---|---|---|
| Range-based | High | Low | High | Medium |
| Range-free | Medium | High | Low | Medium |
| Fingerprinting | Medium | High | Medium | High |
| Cellular | Low | High | High | Low |

Table 4.1: Comparison of Different IPS Localization Schemes

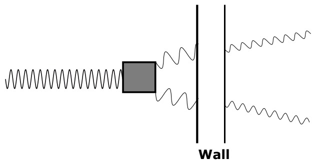



Figure 4.5: Signal Multipath and Attenuation Effects

Trilateration via RSSI has been demonstrated in [207, 208] using values received from location beacons to locate a user holding a Bluetooth enabled device. The proposed systems used laptops or inexpensive Bluetooth dongles as location beacons. They demonstrated adequate performance in controlled environments such as large office spaces and laboratories.

A much different approach that makes use of fingerprinting techniques has also been investigated. Fingerprinting takes advantage of modern sensors being able to access online resources in order to compare and store readings. Data-mining techniques are applied to captured measurements. Fingerprinting involves capturing signal measurements of known locations and storing the features offline. When a node needs to be located, its measurements will be compared to what is captured prior. Data-mining techniques including probabilistic methods, support-vector machines (SVM), $k$-nearest-neighbor and neural networks are then employed to perform the comparison. Fingerprinting has been demonstrated to be promising as it provides good accuracies and does not require deployment of complex hardware. However, capturing enough samples at known locations is a significant undertaking.

Microsoft RADAR [209, 210] makes use of WLan technology and uses fingerprinting methods. RADAR creates a radio map of all access points in an area and corresponding signal strengths. A user's mobile will compare its signal strengths with those on the radio map using a $kNN$ Viterbi-like algorithm. GSM fingerprinting has also been demonstrated to achieve median accuracy up to $5m$ in large multi-floor buildings when using signals from the six strongest cell towers [211].

The drawback of using fingerprinting techniques is that accurate radio maps are required. The radio maps need to be constructed for all areas positioning is required. This increases the cost and effort of deployment, particularity when large areas need to be covered. Furthermore, the maps need to be reconstructed subject to hardware changes. Our approach makes use of easily deployable beacons and neighboring mobile devices to accurately position a node.

Inspired by cellular routing techniques used in mobile cell tower networks, indoor positioning has also been achieved by placing beacons in non-overlapping locations. The limited range of Bluetooth is used as an advantage as a confident location estimate can be achieved when a device is in range. This has been shown to work in a practical system in [212] designed to test context aware phone applications. The authors modified cheap Bluetooth headsets to stay in discovery mode and used them as beacons. An algorithm has also been proposed in how to place beacons in a non-overlapping manner in [213]. While cellular techniques are robust and low-cost, its precision is lower.

While indoor positioning is difficult, many solutions can achieve accuracy within $2m$ to $5m$ in most situations when deployed correctly. This has led to the development of commercial systems available for purchase for specialized applications.

Tadley Systems developed an IPS called Topaz combining the strengths of both Bluetooth and Infrared (IR). Bluetooth is firstly used to determine what room a user is in using a proximity detection algorithm. Infrared is then used to provide an $(X, Y)$ coordinate of where the device is located in the room. Infrared has been shown to be very accurate within a low range and is used by Topaz for refinement.

Bluetooth LE has also been used for indoor positioning. The recent introduction of Apple iBeacon which is built on Bluetooth LE allows for interaction with a smart-phone when a user is in proximity. Systems such as indoo.rs have been built on top of iBeacon and has been demonstrated in practice to work for large scale IPS.

RFID Ultra-Wide Band (UWB) based IPSs have gained popularity, they have proven to be a viable alternative as the UWB frequency has less interference than RF. Ubisense and Snappire Dart have achieved accuracies to the centimeter [214]. The downside is that deployments require many tags, are costly and not widely accessible.

Our approach to Bluetooth localization is completely decentralized and makes use of widely available smartphones and Bluetooth devices.

## 4.3 Problem Definition

In this section, we describe the problem we aim to solve with our unified localization scheme, the APIT range-free algorithm and a method to measure the accuracy of various schemes in order to evaluate their effectiveness.

### 4.3.1 Problem Statement

The problem is to determine the position of a wireless node $n$ in a scene through the use of a localization scheme that makes use of various information available to $n$. A scene can be considered anywhere in space a node may need to be located, examples include a meeting room in a building or in a large lecture theater.

Audible location beacons are considered a special case of wireless nodes in which their location is known ahead of time. They provide location information to querying nodes. For example a location beacon may advertise its $(X, Y)$ position to all wireless nodes in range.

Neighboring nodes surround $n$, however their location is not known. They provide information including signal strength relative to $n$ and audible location beacons in the scene. This information can be provided either by continuously broadcasting it or establishing a communications channel with $n$.

The localization scheme is required to utilize information provided by both surrounding neighbor nodes and audible location beacons to improve estimation accuracy. Localization schemes commonly fall under range-based or range-free approaches. Range-based approaches that use measurements and range-free approaches that use comparisons. By using a hybrid approach that makes use of techniques from both principles, the aim is to determine if better positioning estimates can be achieved by using each method in cases where their strengths can be leveraged while mitigating their weaknesses.

### 4.3.2 APIT

The APIT algorithm was designed to perform well when radio patterns are irregular and nodes are placed randomly. Given three location beacons are present: $A$, $B$, $C$, the aim is to determine whether a point $M$ with an unknown position is inside the triangle $\triangle ABC$ or not. The perfect Point in Triangle Test (PIT) provides a theoretical solution to this problem as presented by the following two propositions.


(a) Proposition I (b) Proposition II

Figure 4.6: P.I.T Propositions

Proposition I: If $M$ is inside triangle $\triangle ABC$, when $M$ is shifted in any direction, the new position must be nearer to (further from) at least one anchor $A$, $B$ or $C$ (Figure 4.6a).

Proposition II: If $M$ is outside triangle $\triangle ABC$, when $M$ is shifted, there must exist a direction in which the position of $M$ is further from or closer to all three anchors $A$, $B$ and $C$ (Figure 4.6b).

A formal proof is available validating the correctness of the propositions. The issue arising from this solution is that the movement of nodes is required and is infeasible to implement. The APIT algorithm approximates the PIT test without requiring any node movement by using information from neighboring nodes. The departure test is run on each neighbor to determine if $M$ is further away from an anchor than a neighbor. The signal magnitude is used to determine this by assuming that the magnitude will monotonically decrease with distance. APIT is formally defined as follows:

Approximate P.I.T Test: If no neighbor of $M$ is further from/closer to all three anchors $A$, $B$ and $C$ simultaneously, $M$ assumes that it is inside triangle $\triangle ABC$. Otherwise, $M$ assumes it resides outside this triangle.

The algorithm runs APIT on every combination of triangle that can be formed with the location beacons. APIT aggregation is then run when each individual APIT test is finished. By using a grid SCAN algorithm, each grid cell is incremented or decremented based on whether an intersecting triangle passes or fails the APIT test. The grid area that has the maximum values is used as the area that is the most likely where a user is located, the center of gravity of this area is considered to be the location estimate.

### 4.3.3 Mean Estimation Error

To measure the accuracy of our approach and other various approaches, the estimation error is calculated by calculating the Euclidean distance between the actual user position and the estimated position. Accuracy can vary depending on room position, to account for this multiple measurements are taken to determine the mean estimation error.

$$MeanEstimationError = \frac{\sum_{i=1}^{n} EstimationError_i}{n}$$

The equation above is used to calculate the mean estimation error in our testing where $i$ is the sample index for where the actual user location is and $n$ the number of samples taken in a setting.

## 4.4 Our Unified Approach

In our approach, techniques are drawn from both range-based and range-free methods. The unified approach we propose makes use of the range-based trilateration using signal magnitudes to calculate distances combined with the range-free APIT algorithm [193] which is used for further refinement.

Initially the three closest beacons are determined by selecting the top three audible location beacons with the greatest signal magnitudes. Distance estimates are calculated to each of these three location beacons via a signal propagation model. In the case presented in Figure 4.7, the location beacons are $P1$, $P2$, $P3$. The node the algorithm is attempting to position is assumed to be somewhere within these beacons, the position need to be narrowed down further.

Trilateration is then performed using these estimates by placing three circles around each location beacon using the position as the center point. The radius of the circle is set as the distance calculated initially. It can be seen that these circles intersect at a point, the center of gravity of the intersection is considered the location estimate. The resulting location from the trilateration is then used as the center point to create a bounding box $Bm$ which is indicated by the red box at the intersection of the three circles. At this stage, it is assumed that the location estimate could be refined further within the bounding box

The APIT algorithm is applied to the entire scene. Triangles are formed between all combinations of audible location beacons creating a geometric overlay. For each formed triangle, it is determined whether the node that needs to be positioned is inside or outside each triangle. This is determined by running an APIT test which examines the surrounding

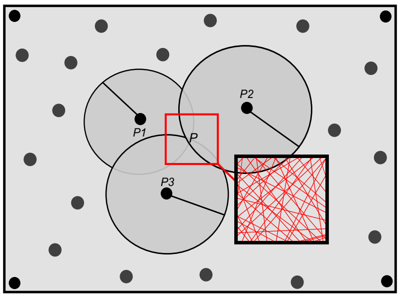



Figure 4.7: Unified Approach to Localization

neighboring nodes to determine if any are further or closer to all three location beacons that form a triangle. A grid is then overlaid on top of the scene for which APIT aggregation is run.

Triangles within the red bounding box are clipped using the Sutherland-Hodgman algorithm [215], this will return a list of polygons. This can be seen in Figure 4.7, all the triangle lines forming the initial triangles terminate at the edge of the box. All polygons inside the bounding box that intersect with any grid cell with the maximum SCAN index value are taken and the intersection between these polygons is found.

The center of gravity of the resulting polygon is used as the new refined positioning estimate. This has the effect of refining the initial position obtained by the range-based to be closer to the actual user position using information provided by surrounding neighboring nodes.

### 4.4.1 Unified Algorithm Walkthrough

In this section, we detail each steps taken by our algorithm to locate a node. These steps need to be run consecutively as the range-based method is used to establish a range where the node is located while the range-free method is used to refine the position within that range. This does not work as intended in reverse as range-free methods are susceptible to accumulating error in large spaces.

Our unified approach performs the following:

1. A node that requires positioning scans for audible location beacons, it can sense and requests their location and signal strengths. The received data is stored in a table and sorted by signal strength (Figure 4.8). The signal strengths provides a relative indication of node proximity to the fixed location beacons.

| | $X_m$ | $Y_m$ | $RSSI_n$ |
|---|---|---|---|
| A | 0 | 0 | -63 |
| B | 0 | 10 | –60 |
| C | 10 | 10 | -57 |

Figure 4.8: Storing Values Sensed

2. The three closest location beacons are selected using the RSSI value as an indicator of distance. The algorithm assumes the user is somewhere within the triangular region connecting the location beacons (Figure 4.9) further narrowing down the node position.

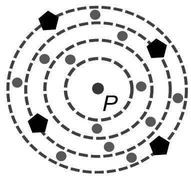



Figure 4.9: Closest Bluetooth Nodes are Found

3. Distance estimates between the user node and location beacons are calculated using the path-loss equation described in Figure 4.10. The RSSI is set as the signal strength of the location beacon, $RSSI_0$ needs to be known ahead of time, this information can be derived ahead of time for different receivers. The distance estimates provide spatial information of node positions relative to surrounding nodes.

$$RSSI = -10 \times n \times \log_{10}(distance) + RSSI_0$$

| | |
|---|---|
| $RSSI_0$ | RSSI value when receiver is 1m away |
| $n$ | The rate at which path loss increases |

Figure 4.10: Path Loss Equation

4. Trilateration is performed using the distance estimates to obtain an approximate location. This is performed by drawing three circles from the center point of each respective node, the radius is the distance estimate. The location estimate is the center point of where these circles intersect. This location is assumed to not be precise, however close to the user and other nodes surrounding the user (Figure 4.11).

5. A bounding box of $B$m is created using the rough user position as the center point, this results in a region with an area of $A^2$. A range-free technique is then employed within the bounding box (Figure 4.11). The positioning estimate is constrained within $B$.

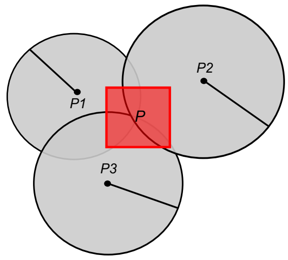



Figure 4.11: Bounding Box Surrounding Approximate Location

6. The node builds a table combining the signal strengths for each beacon, its location and the signal strengths of each beacon from the perspective of neighboring nodes (Figure 4.12). This information is later used to refine the positioning estimate within $B$.

| | $X_m$ | $Y_m$ | $MyRSS$ | $RSSI_1$ | ... | $RSSI_n$ |
|---|---|---|---|---|---|---|
| A | 0 | 0 | -63 | -72 | ... | -41 |
| B | 0 | 10 | –60 | -66 | ... | -45 |
| C | 10 | 10 | -57 | -60 | ... | -43 |

Figure 4.12: Merging Tables of Sensed Values Across Devices

7. Triangles are generated using every combination of audible beacons, thus leading to $\binom{n}{3}$ combinations. These triangles are then used in APIT range-free localization (Figure 4.13). These triangles are used to narrow in on the location.

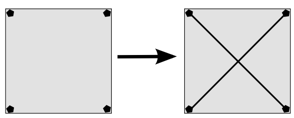


Figure 4.13: All Combinations of Triangles are Generated

8. Each column is evaluated using APIT; if a neighbor node exists with consistently larger or smaller RSSI values then the user assumes a position outside the triangle, otherwise inside (Figure 4.14). Triangular areas where the node is not found to be within can be disregarded, thus further refining the estimate.

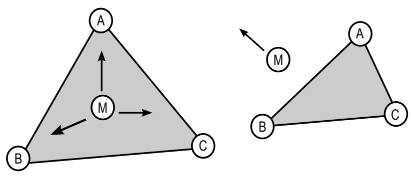



Figure 4.14: Determining if the Node is Inside or Outside the Triangle

9. Step four is repeated for every combination of three beacons (Figure 4.14).

10. For each individual APIT test, if a node is estimated to be inside the triangle, the scan index value is incremented, otherwise if outside the value is decremented. The scan index values are maintained in a table with a reference to each triangle (Figure 4.15). This scoring is used as a proxy of the likelihood of nodes being within the triangular areas.

| | |
|---|---|
| Triangle A | 5 |
| Triangle B | 2 |
| Triangle C | -5 |

Figure 4.15: All Combinations of Triangles are Generated

11. A grid with a side of $0.1 \times RoomWidth$ is overlaid. The scan index of each triangle in which the point resides is aggregated to the grid cell (Figure 4.16). The grid allows for taking into account the overlap between triangular regions and allows for further areas that do not overlap to be disregarded from the positioning estimate.


Figure 4.16: SCAN Grid

12. Triangles are clipped using the Sutherland-Hodgman algorithm [215] and the bounding box from the initial trilateration, this results in clipped polygons (Figure 4.17). Any triangular areas that are not within the bounding box are not considered in the final positioning estimate.

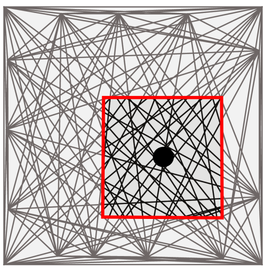


Figure 4.17: Polygons Clipped

13. The center of gravity of the resulting polygon is then found by taking the intersection of polygons that share the maximum scan grid value, and used as the positioning estimate

(Figure 4.18). The center of gravity of the overlapping polygons is the most likely to be the correct positioning estimate as the node was found within all the overlapping regions.

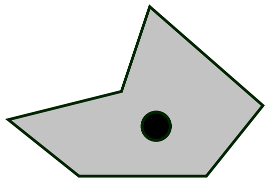


Figure 4.18: Center of Gravity of Polygon

## 4.5 Virtual Beacons

In localization schemes, a sufficient number of location beacons are required to achieve accurate positioning. The amount required depends on the technology being used. For example, if the range of a beacon is only $10m$, then a sufficient number of beacons need to be placed every $10m$ to achieve accurate positioning within the area.

In the case of the APIT range-free method, additional beacons placed across an area leads to added combinations of possible points in which triangles can be formed. Since placing many real beacons may not be feasible, we propose the use of virtual beacons.

### 4.5.1 Signal Magnitude of Virtual Beacon

Virtual beacons derive proxy RSSI values using real beacons as a point of reference to then infer what an RSSI value in a nearby location would be. Figure 4.19 illustrates the flow of how RSSI values are assigned. These values are derived using the path loss equation which is detailed in [216].

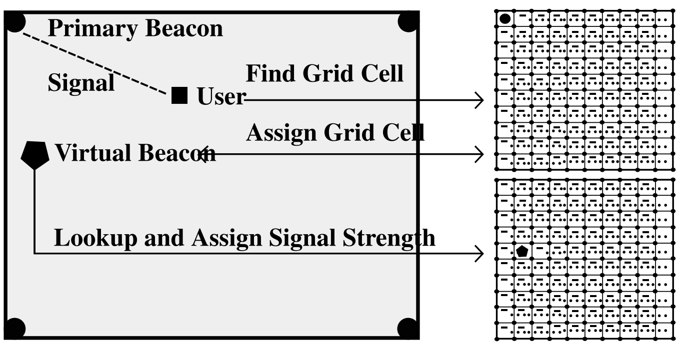



Figure 4.19: Calculation of Virtual Beacons Signal Strength

Each real beacon in a scene maintains a grid separated by $Xm$, each grid point contains an RSSI value calculated from the path loss equation using the Euclidean distance from the grid center point and the beacon location.

The reference beacon is selected based on which real audible location beacon has the strongest signal. To assign an RSSI value to the virtual beacon based upon user location, a lookup is performed to find the cell that is the closest match based off the primary beacon RSSI. The resulting grid cell is then used to perform a reverse lookup on the virtual beacons grids.

The resulting RSSI is assumed to be the RSSI from the user's perspective to the virtual beacon. The key to placing virtual beacons is to place them not too far from the real beacons and inferring its RSSI value based off the closest real beacon.

## 4.6 Experiments

To evaluate our approach, we developed a Bluetooth simulator in Java for the purposes of testing and comparing different localization schemes. The simulator allows for Bluetooth location beacons and Bluetooth nodes to be placed arbitrary in a room.

### 4.6.1 Indoor Scenario

The indoor scenario we considered for testing was to position a node in a $10m$ x $10m$ room (Figure 4.20). We considered this a good candidate room size as many Bluetooth devices have a maximum range of $10m$. In addition many rooms in indoor settings rarely have an area larger than $10m^2$ unless considering larger floor spaces like hall ways and office spaces. In these cases additional location beacons can be deployed. Provided beacons are placed in every room where node positioning is required and additional beacons are placed in spaces larger than $10m$, our approach scales.

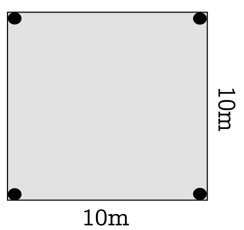



Figure 4.20: Indoor Positioning Scenario

### 4.6.2 Positioning Estimate Locations

Location estimates can vary in accuracy depending on where the node that needs to be positioned is located in the room. To account for this, location estimates were taken at multiple locations to calculate the mean estimation error as described in Section 4.3.3.

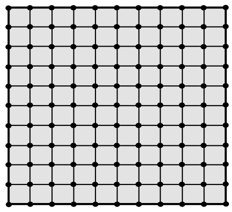


Figure 4.21: Overlay Grid

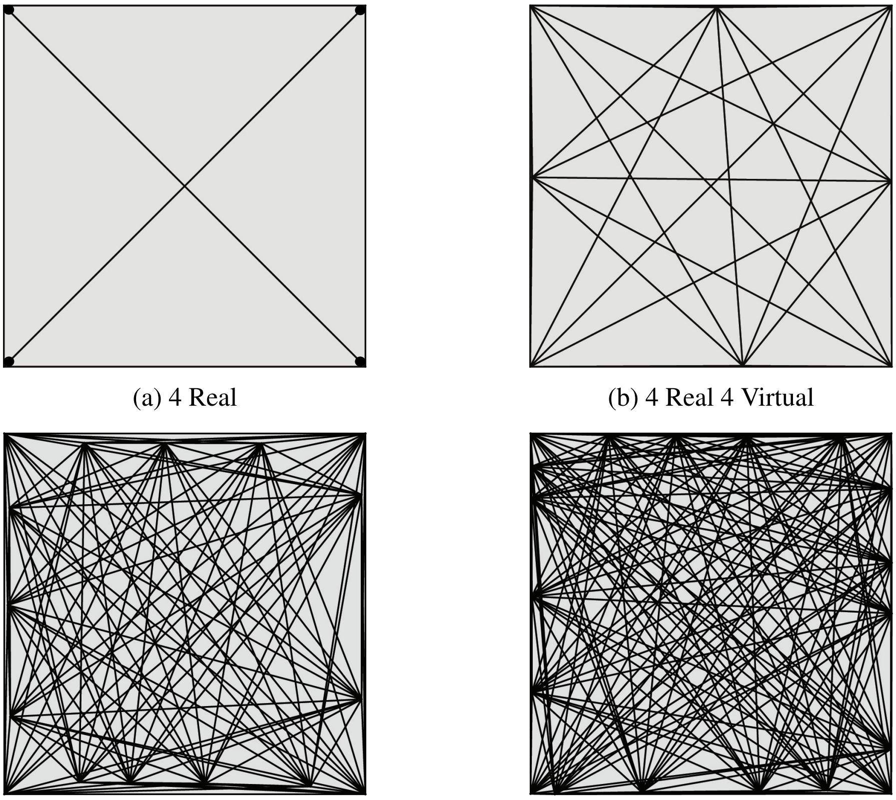
By overlaying a grid on top of a room with a grid side of $1m$, an estimation is performed at each cell as highlighted by the black circles in Figure 4.21. In the indoor setting we tested, 100 location estimates are captured across the entire $10m$ x $10m$ space effectively covering every position a user can stand in a room.

### 4.6.3 Location Beacon and Node Placement

Location beacons were placed in six different configurations starting with 4 real beacons and gradually increasing to 20 beacons in total as listed in Figure 4.22. In a real scenario, placing more than four beacons in a $10m$ x $10m$ room may not be practical to deploy or maintain, for this reason we considered the use of virtual beacons. Four real beacons were always placed in the corners of the room, the remaining beacons are virtual. The real beacons are used to bootstrap the virtual beacons values.

Neighboring Bluetooth nodes surround the user node the localization scheme attempts to position. The ubiquity of mobile smart phones makes it realistic to consider that there would be devices nearby that could be used as additional nodes for refinement by a range-free component. In our experiments, we increased the neighboring nodes from 1 to 15 and test if an increase leads to better positioning accuracies.

Two configurations for placing neighboring Bluetooth nodes surrounding the user Bluetooth node is considered. Random placement and Uniform placement. In random placement the $(X, Y)$ coordinates are generated randomly. In Uniform placement neighbor nodes are increased perfectly spaced centered vertically and horizontally.



(a) 4 Real

(b) 4 Real 4 Virtual

(c) 4 Real 12 Virtual

(d) 4 Real 16 Virtual

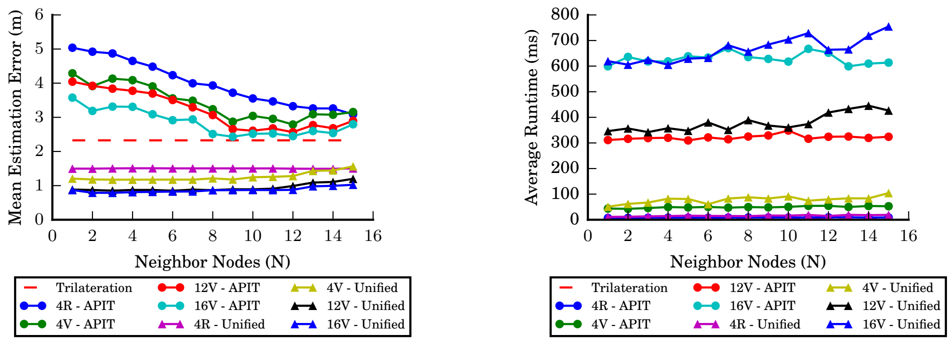
Figure 4.22: Beacon Configurations Used in Our Evaluation

### 4.6.4 Signal Strength Simulation

For each Bluetooth node, perceived RSSI values are calculated for each location beacon that can be sensed via the path loss equation by setting the distance value to be the Euclidean distance between the location beacon and the respective node.

To add some interference, the propagation constant was set to $n = 2.2$. As evidenced in ZigBee wireless networks and transceivers [217], this propagation constant represents what would occur in a retail store. This simulates what happens in real scenarios where signal attenuation occurs due to various objects in a room.

Through experimental evaluation of multiple real Bluetooth devices, we found the RSSI value tends to be -51 when placed $1m$ from one another. Thus, $RSSI_0 = -51$ was set in the simulator when calculating RSSI based off path-loss. When taking range-based estimations, the localization algorithm assumes $n = 2.0$ as the exact environment in which we are operating is not known.




(a) Mean Estimation Error (b) Avg. Runtime

Figure 4.23: Results When Using Virtual Beacons, Neighbor Nodes Arranged in a Uniform Placement

### 4.6.5 Evaluation

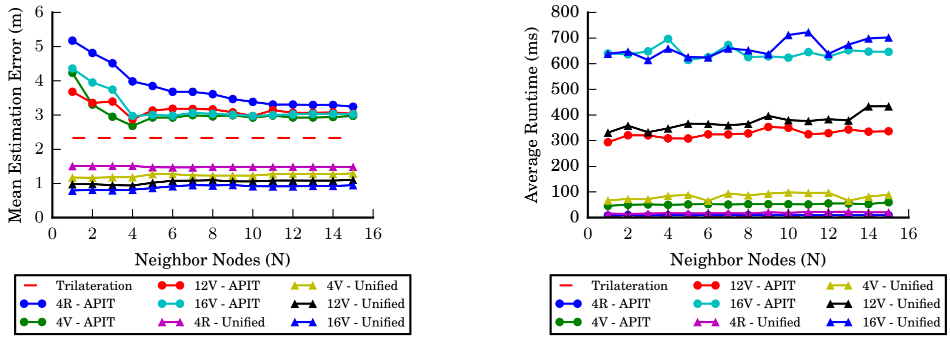
In our evaluation we tested RSSI Trilateration and APIT isolation in addition to our unified approach to clearly demonstrate the difference in performance. The location beacon configurations tested are displayed in Figure 4.22 are as exported by our simulator. Neighboring nodes were placed in both a random and uniform manner to study the effects of whether or not the positioning of neighboring node locations makes any significant difference.

It can be seen in Figure 4.23a and Figure 4.24a that the unified approach performed better than when using either RSSI Trilateration or APIT in isolation. The red dashed line indicates RSSI Trilateration, because neighboring nodes and virtual beacons are not considered by this method, it always produces a consistent MSE of $2.30m$ in every case. Our unified approach achieves a MSE of just under $1m$ in some cases.

To put the improvement in estimation error into context, current methods achieve an accuracy of $2m$ to $3m$ while our method improves this to under $1m$. In a scenario where a user needs to be located holding a mobile device in a building, our approach can differentiate between two people standing next to one another while current methods may confuse the two people.

The average runtime our approach as shown in Figure 4.23b and Figure 4.24b indicates that the positioning estimates can be derived in under a second. Communications costs remain low since the neighboring nodes and positioning beacons only send signal strength information with the node that needs to be positioned only once.

Initially in our preliminary experiments, location beacons were placed in a perfectly




(a) Mean Estimation Error

(b) Avg. Runtime

Figure 4.24: Results When Using Virtual Beacons, Neighbor Nodes Arranged in a Random Placement

uniform manner along the edges of the room. We achieved much better results when the placement is less symmetrical due to the APIT test better predicting if the user node is inside or outside a triangle. Thus, while additional beacons were placed in a near symmetrical manner, they are not perfectly aligned.

Increasing neighboring nodes leads to improved performance in a logarithmic like trend when only using APIT. However in the Unified approach the effect of increasing nodes is negligible. This is because the bounding box restricts the search space to the point where looking at just one neighbor node is enough to refine within the box further. Our results suggest, there only needs to be at least 1 neighbor distributed around the room to achieve high accuracy.

The baseline to improve upon when using virtual beacons is to compare the results to when using only four real beacons in isolation. Virtual beacons are only considered for use in the range-free APIT component of the algorithm. This is due to range-based method being much more sensitive distance errors, range-free methods are more robust as they do not directly rely on measurements.

When placing neighboring nodes uniformly as opposed to randomly, the trend of improved performance attributed to the increase in virtual beacons is slightly more prevalent. It can also be seen that uniform placement of neighboring nodes eventually leads to the performance of APIT converging with the performance of trilateration. Adding virtual beacons while leading to increased performance, the performance improvements plateaus once many are added in the space.

## 4.7 Conclusions and Future Work

In this chapter, we proposed a unified approach to localization and introduced the concept of *virtual beacons* to further refine location estimates. The key contribution of this work is using a normally considered fine-grained range-based methods to get a rough estimate and a course-grained method to refine further within smaller search space can improve positioning results.

Initially, it may not seem obvious to combine a fine-grained localization method with a coarse-grained localization method to improve the accuracy of positioning estimates. However, in practice radio maps are not usually available and performing localization simply via triangulating from the use of Bluetooth location beacons is insufficient due to signal multipath and propagation effects. Applying a coarse-grained approach to refine the position within an estimated boundary based on the position yielded by the fine-grained method can be used to further improve the accuracy of the positioning estimate.

The accuracy of range-based methods suffer when signal propagation occurs, however by using the three closest beacons based on the greatest signal strength, a user will be within the triangular area. The range-based method does not take into account any neighboring nodes to refine further. This is where range-free methods can be used. The magnitude of signal strengths of each node in an area to the anchored location beacons to further refine position. Because they work using binary decisions, they suffer in cases where a node is on an edge. Furthermore they also tend to favor location estimates closer to where the concentration of nodes is.

Virtual beacons were also introduced as a means of further refinement. They derive a signal magnitude from a primary beacon and allow for a finer spatial resolution. The signal magnitude of a virtual beacon is derived by the primary beacons maintaining a table of RSSIs derived via path loss equation. A virtual beacon looks up the table of the closest real beacon to assign a signal magnitude.

Our experimental results indicate that using our unified approach produces more accurate location estimates than when using either approach in isolation. Furthermore when used correctly, virtual beacons can improve performance and provide a comparable performance to real beacons when a small number is derived from each real beacon.

At present we are in the process of implementing our IPS based on our proposed unified localization scheme. The IPS is built on top of the Android platform and will be evaluated by taking location estimates at known locations and comparing where the IPS estimates the user to be. The IPS will be deployed in a campus setting, a user study will also be performed to test its usefulness as an indoor navigation tool.

In future work we plan on working on an algorithm to determine the optimal configuration of beacons in larger spaces when using our localization scheme. We also wish to investigate the effects of virtual beacons further both determining what their optimal configuration is and if they work to improve performance in other localization schemes.

# Chapter 5

# Mining Encounters of Smartphone Users in Real-Time

Location information can be used to determine the high level activities of a user. In this thesis, we demonstrated how to position a smartphone user's location indirectly via analyzing the query results or analyzing Bluetooth signals. In this chapter, we reveal how direct location updates can also indirectly reveal encounters a smartphone user may be having with other individuals. Recent advancements in data mining coupled with the ubiquity of mobile devices has led to the possibility of mining for events in real-time. We make the assumption that a service provider has direct access to continuous query requests made by a smartphone user. Many location based apps and services make use of continuous queries, thus providing a reliable snapshot of user locations in near real-time. Continuous requests provide an accurate proxy of where a user is located. With this knowledge we reveal how encounters with other smartphone users can be detected in near real-time.

As people travel, they may have encounters with one another. We are interested in detecting the encounter patterns of traveling individuals at the exact moment in which each of them occur. We introduce the problem of mining for an individual's encounters. A simple solution is to use a nearest neighbor search to return potential encounters, this results in slow query response times. It is significantly more efficient to check just for neighbors that are in predefined proximity instead. Current techniques miss potential encounters that happen quickly because the evaluation time is too long for city scale problems, thus the benefits of continuous mining will not be fully realized.

---

**This chapter is based on**
**Mining City-Wide Encounters in Real-Time - SIGSPATIAL 2016**

To mine for encounters in real-time, we introduce a new algorithm that is efficient in capturing encounters by exploiting the observation that just the neighbors in a defined proximity needs to be maintained. Our evaluation demonstrates that our proposed method mines for encounters for millions of individuals in a city area within milliseconds.

A user may be happy to reveal their location but may not want to reveal their encounters. A service provider that many smartphone users make use of such as a popular social network have access of location updates for many users. If users are having an encounter and are using the same social network, we can determine if they are in proximity. However, in this chapter we demonstrate how a user's location can also indirectly reveal their encounters.

## 5.1 Introduction

Vast quantities of spatial data is generated by mobile smartphone apps. Captured data in turn is sent to online services for processing for the purpose of providing additional functionality. For example, tagging a place on a map or routing indoors based on GSM signals and user location [211].

The occurrence of encounters between people are possible when they are within proximity to one another. Many individuals keep their smartphones present with them at all times or at least within the same room [218]. This provides an approximate location of where the individual is located. With spatial data captured from mobile devices, it is possible to mine for potential encounters by checking for individuals that are in proximity to each other via performing queries on the spatial data. Many devices now send continuous positioning estimates, allowing for the detection of encounters in real-time.

Mining for encounters has many practical applications. Benefits to an individual include suggesting potential places for meeting participants, automatically finding more suitable meeting times and suggesting new social connections by finding nearby friends to join an open meeting. Organizations will also benefit from real-time mining of encounters, for example in law enforcement it can be used to monitor exchanges between a predefined set of persons of interest.

Early apps that used location often updated user positions during the app being in focus. For example, many users access and update social networks frequently. Modern smartphone apps now send continuous updates to these services of user positioning estimates captured by the device. With the data generated and collected, it is possible to determine if users are in proximity of each other. Social networks such as Facebook continuously collect location updates.

Current apps make use of coarse grain proximity, insights derived from using fine grained proximity still need to be explored. The concept of using proximity in mobile apps has found use in mapping and social network applications. Google Maps provides a list of the closest POIs and Instagram lets you tag photos to places. Social networks have recently started to display users in coarse proximity where a relationship exists. At present, fine-grained proximity information and the insights it provides such as encounter patterns is not being used to improve services.

Spatial indexes are used by spatial databases to optimize queries such as nearest neighbor search. Many existing spatial databases used by an LBS cannot mine for user encounter patterns in real-time. Commonly used methods for indexing are R-Trees and QuadTrees. An issue arises when using these methods for moving objects since the index is required to be updated right before a query is initiated. Indexes for moving objects have been proposed such as the TPR tree [219] and the $B^x$ tree [220, 221]. With these methods it needs to be verified if an object is actually within a query region as variables such as speed of a user or in some cases trajectory are assumed. Another drawback with using these methods for our problem is as time increases, the number of possible locations a moving object can be located based on the velocity vector increases. This results in having to process more combinations to check who is actually within proximity to whom as time increases if the index is not updated frequently. These indexes are used to execute queries such as $k$-NN.

Applying a $k$-NN query to check for individuals in proximity is not an efficient method to mine for encounters. We do not need to determine the nearest neighbors but rather the individuals located within a close radius. For example, it is possible for a $k$-NN query to return an object that is not in proximity if no immediate surrounding objects are found.

We define the constraint nearest neighbor ($c$-NN) query to return points that are within a defined close proximity as our first step. Given the $c$-NN constraints, it is possible to exploit proximity to build an efficient method that is used to return the results by taking into account the redundancy of not having to search for points not in the immediate vicinity.

Proximity is a constraint required for individuals to be having an encounter. We exploit this property for maintaining an index that stores points that are close to each other in a given area. Our novel approach is based on using a dynamic grid that maintains and stores the cells that individuals have not moved for a specified time period. At each interval, an efficient scan is executed to determine what user locations intersect with grid cells. For example, people that are in the same grid cell may be potential candidates for encounters. We developed a simulator that creates an environment of people traveling in an inner city area and meeting and having possible encounters with other people. The simulator uses real geospatial data provided by Open Street Map (OSM).

To the best of our knowledge, this is the first work that focuses specifically on encounters. We demonstrate our novel algorithm to mine for these associations in real-time. Our algorithm runs in the milliseconds for two million people in the Melbourne inner city. It is significantly more suitable for real-time mining than conventional techniques, which run orders of magnitude slower.
Our main contributions are as follows:

- Define the mining for encounters problem
- Provide a novel algorithm to mine for encounters
- Evaluate our algorithm and compare it with other possible techniques

## 5.2 Background

A significant amount of research has been directed at the storage and retrieval of spatial data. In this section we provide an overview of certain techniques and how to apply them to our problem. We found that these techniques are not suitable for our setting.

### 5.2.1 Static Spatial Indexes

Modern databases use spatial indexes to improve the query execution speeds. The use of an index improves performance by graphing points or regions in a manner by exploiting various spatial properties for improving query execution time. Two techniques in widespread use include the QuadTree [222] and R-Tree [223]. The index to be used is dependent on the problem.

For our problem, static indexes need to be built right before a query is initiated to capture individuals that are having a potential encounter. Repeated nearest neighbor queries is required to be run for each user to find those in proximity. Fast methods to perform NN queries using static indexes have been proposed [224, 225]. While static indexes is generally quick to build, if it has to be updated continuously then they do not scale well as the index needs to be rebuilt every time a user moves, in addition a NN query has to be repeated for all objects in the database.

Range queries can also be performed efficiently using spatial indexes to retrieve neighbors that are within a specified proximity. Instead of performing a direct NN query that would retrieve the closest neighbors irrespective of the distance, a range query can be performed. When using a range query, a rectangle with the center point being the query point is generated. This rectangle can then be used to find any neighboring points that intersect with it.

Another approach is to use hashing. For example, the service geohash.org provides a short string given a specific geographic coordinate. The key idea is to use Z-Order curves [226] to reduce the data into a single dimension. The advantage of Geohash is that it provides a convenient representation that allows to sacrifice precision for the length of the hash string. Reducing precision is used as a means to bucket a collection of locations. Points that generate the same hash would naturally be in close proximity in most cases, by observing the prefix, a quick nearest neighbor search can be performed. With this method, hashes need to be refreshed at query time leading to scalability issues as well. None of the methods we have seen so far also consider the concept of proximity in the index or query execution design.

### 5.2.2 Moving Object Indexing

Indexing moving objects is an active area of research. The need for efficient techniques is in high demand with the proliferation of wireless devices. Velocity based approaches use speed and trajectory information to reduce the updates required to the underlying structures. This leads to the index not having to be rebuilt each time a query is required.

The TPR tree is an extension of an R-Tree designed to handle queries in the time dimension. It takes into account current and future positions of moving object data [219]. The TPR tree uses a minimum bounding rectangle (MBR) in addition to a velocity vector to model the object moving in time. The time a query is initiated is captured. The area is then calculated to determine if there is an overlap with the query region. We can use this instead for a solution to our problem, however a large area will be interpreted at query time due to possible locations a moving object can be located at as time increases. In particular, for a large set of neighbor investigations even a minor change in the size of the area under investigation will lead to large execution times.

To overcome the high cost in node splitting exhibited in TPR Trees, the use of dynamic $B$ trees [220, 221] has been explored. The $B^x$ tree maintains a directory in each node containing the id, velocity, mapping and last update time of an object. The mapping of two dimensional points into a single dimension is achieved using Z-order curves [227]. Linear interpolation is used to project the position of an object at a future time.

The $B^x$ tree requires an object moves in fixed velocity. When capturing location tracks in real-time, we do not know the direction a user travels in. We found this approach did not work well for mining encounters. Using this method would require many frequent updates that would lead to little benefit against using a static spatial index.

Certain Quadtrees [228, 229] can be used to index moving objects as demonstrated in [230]. By using loose quadcells, an object can move and not need to be reinserted provided the new location is still within the quadcell. There is an expansion factor which defines how much a quadcell can enlarge. Thus, these methods work well if it is known ahead of time what area an object can cover and the object does not move relatively far from the initial position. The concept of loose quadtrees can be applied to the mining encounter problem, however since a user can cover positions in a really large area, many frequent updates would be required. There is no defined bounds where a person may move in a city area. These approaches also do not consider any tolerance constraints nor the fact that we are interested in ANN query types rather than k-NN.

### 5.2.3 k-NN Queries

Mining for encounters is related to the nearest neighbor problem. The nearest neighbor problem being for a set of points $S$ and a query point $q$, what is the closest point from $S$ to $q$ [231]. When mining for encounter patterns, the location of each person can act as the query point $q$, the nearest neighbors of each person can then be tested if they are in proximity to the person to determine if they are having a potential encounter with each other.

One immediate issue with applying $k$-NN queries to mining for encounters is neighbors that are not in proximity would also be returned. It has to be checked that the users returned are actually in proximity. A monitoring continuous $k$-NN approach [232] uses a grid. This approach does not work well in our application domain as building and maintaining an entire grid over a large area is costly. Setting the grid cell size too small leads to long execution time for a large city. Setting them too big causes too many hits. The authors also proposed an incremental approach to updating index which was only demonstrated to work well for low velocities.

One method proposed in [233] uses an R-Tree to compute k-nearest neighbors for a trip efficiently. The method assumes that the line segment a user is traveling needs to be known ahead of time and neighbors are not moving. This technique cannot be applied to our mining for encounters as we do not know where the user is traveling.

An expansion of [233] was proposed in [234] to take into account moving objects for kNNs by using a TPR-Tree as the underlying structure. Further methods making use of a TPR tree were proposed in [235]. We found that TPR trees do not work well for our application domain as stated in the previous subsection. This same observation was also noted independently in [232].

### 5.2.4 ANN Queries

When mining for encounters, we actually want to find possible occurrences for every person. The problem can be represented as the all nearest neighbor (ANN) problem that states for all $N$ points, find the $k$ nearest neighbors in a certain proximity for each of the $N$ points. Efficient solutions for the ANN problem have been proposed with the basic premise being to exploit redundancy [236, 237].

In [236], a PR QuadTree with a fixed cell size is used to store points, Each cell is labeled using the total of points stored. This leads to a reduction of the search space to be based on size of the quad cells. The scan time is bounded by the depth of the tree. An improved variant [237] proposed the use of box cells that are dynamic cells around groups of points reducing the query time to be bounded based on the number of points. This approach assumes that all information is stored in main memory.

A more recent approach exploits the distance between blocks storing the points [238].

The use of a high level structure such as a QuadTree is used to generate the containers. Each structure containing points is assigned a label indicating the number of points it contains. The $MinDist$ and $MaxDist$ between the blocks in the structure is then calculated. When finding the neighbors required, it can be quickly seen which blocks to select to get the desired number of points. The ANN approaches assume that the points are not moving, when points move then the structures in most cases needs to be recalculated.

There has not been much attention focused on continuous ANN. One approach recently proposed in [239] presented an algorithm that uses cell tower radius as a proxy to reduce the search space. This is not appropriate for our problem since it would yield too many results that would manually need to be checked if they are within close proximity since a cell tower radius can cover a large area.

Our algorithm makes use of redundancy in both the time and space domains to reduce computation. As a result there is far less points that need to be stored in the index and the number of combinations that need to be processed.

## 5.3 Problem Definition

A potential encounter can occur when people are in close proximity. It is assumed people are within proximity when the distance between them is within $\epsilon$ distance. We define the constraint nearest neighbor (c-NN) query to return objects in proximity at a given a location.

People that are being monitored for encounters can be represented as a set. Let $q$ be the number of people in the search space and $P$ be a finite set of people. We have $P = \{p_1, p_2, p_3, ..., p_q\}$. People may either move or remain static as time increases, an encounter may occur at a location when people move within proximity to each other.

A Euclidean space $\mathbb{R}^2$ is used to define location points where people are positioned at a particular point in time. Let $L$ be the set of locations where a person can be located as well as an encounter may occur. We have $L = \{l_1, l_2, ...\}$ where $l_i \in \mathbb{R}^2$. The moment in time that a person is at a location or having an encounter is captured by a timestamp. Let $T$ be a set of timestamps used to indicate when encounters occur and where a person is located at a particular point in time defined as $T = \{t_1, t_2, ..., t_j\}$ for $1 \leq j < \infty$.

To determine if people are having an encounter, the locations of the people in question needs to be retrieved in order to see if they are in proximity to one another. A person $p_a \in P$

is at a location $l_m \in L$ is returned by the function $Loc_t(p_a)$ at a point in time $t \in T$ is returned as follows

$$\text{Loc}_t(p_a) = l_m \text{ where } t \in T, p_a \in P, l_m \in L$$

Assume that a constant $\epsilon$ is used as a measure of proximity between locations. Consider the two locations $l_m \in L$ and $l_n \in L$, they are in $\epsilon$ proximity if the following condition is satisfied

$$\text{Dist}(l_m, l_n) \leq \epsilon$$

When monitoring locations of interest, surrounding locations need to be determined. Consider the location $l_m$, let $R(l_m)$ be the set of locations that are within $\epsilon$ distance defined as

$$R(l_m) = \{l_o \in L \mid \text{Dist}(l_m, l_o) \leq \epsilon\}$$

The set of people at location $l_i \in L$ and in the region surrounding $l_i$ at a certain time $t \in T$ is returned as follows

$$\text{LocP}_t(l_i) = \{p \in P \mid \text{Loc}_t(p) \in R(l_i)\}$$

For people to have an encounter, it is required to determine if they are in proximity to each other. This is achieved by taking the person in question and determining if their location is within close proximity to others in the set. Let c-NN$_t(p_a)$ be the set of people in proximity to $p_a$ defined as

$$\text{c-NN}_t(p_a) = \{p \in P \mid \text{Dist}(\text{Loc}_t(p_a), \text{Loc}_t(p)) \leq \epsilon\}$$

Encounters occur at a given time when people are in proximity. The combination of people meeting can be derived using a power set ($\mathcal{P}$). Let $E_t$ be the set of all people in proximity that have encounters, defined as

$$E_t = \{\mathcal{P}(\text{c-NN}_t(p)) \mid p \in P\}$$

In cases where encounters need to be found at a given location, this can be defined as follows

$$E_t(l_i) = \{\mathcal{P}(\text{c-NN}_t(p)) \mid p \in P,\ p \in LocP_t(l_i)\}$$

## 5.4 Mining for Encounters

In order to mine for encounters efficiently, we propose the use of proximity and time constraints to significantly reduce the search space. The aim is to use a c-NN query to quickly find all the individuals that are within proximity. We propose the use of a spatial index that exploits these properties. Building a spatial index for the purposes of answering (c-NN) queries allows for the fast determination of $E_t(p)$ where $p \in P$.

We refer to our spatial index as TimeGrid. Our index marks cells as active when they contain more than one person or in the case of a cell containing one person, another person can be found in an adjacent cell. We only consider encounters that occur in the one region and do not move from that region. The index also imposes a time constraint requiring that the people are in the same cell for a specified period of time. The TimeGrid index can satisfy c-NN queries without having to perform an exhaustive search.

Our spatial index is based on using the properties of a grid to index people close to one another. Given a grid cell, people indexed in the same cell must be in proximity to one another while people in adjacent cells may also be in proximity. Thus, the query is already being answered in each timestamp by means of updating our index. Updates are also done in an efficient manner as presented in the next section.

### 5.4.1 Indexing and Mining using a Grid

A potential encounter can occur when people are in close proximity. It is assumed people are within proximity when the distance between them is within $\epsilon$ distance. We define the constraint nearest neighbor (c-NN) query to return objects in proximity at a given location.

In order to search for people that are within proximity to one another quickly, a grid structure can be constructed and used as a spatial index. In a grid, the distance between two objects can be approximated as the distance between the two grid cells, where the objects are located. Constructing a grid removes the need to calculate the exact distances between all possible pairs of points. In other words, people in proximity can be found by considering all the people that are in the same cells or surrounding cells if the grid is constructed and interpreted carefully.

We propose TimeGrid, a spatial index that takes into account time as well as space. Since in many cases it may be only of interest to capture encounters that last a certain amount of time, the minimum time an encounter has to last before it is reported is defined as $\tau$. The underlying data structure is required to maintain a list of people that fall within the grid cell for $\tau$ seconds. Given the cells are set to have a side of size $\epsilon/\sqrt{2}$, a person that is in the same grid cell for $\tau$ seconds with another person indicates a potential encounter.

Each person $p_a \in P$ is positioned at a location $l_i \in L$, all locations are within $\mathbb{R}^2$. The space itself can be partitioned into a grid which can then in turn be used to index each person in the set $P$. We define a grid overlaid with each cell to be of size $\delta = \epsilon/\sqrt{2}$ in each dimension. Thus, locations within a grid cell would be within a maximum of $\epsilon$ distance from one another. The grid over the entire space is defined as $G$ with each cell having a side of $\delta$. Each person in $P$ is indexed to determine which grid cell they reside in and represented in a set. It is possible to map the set of people to a grid cell ($P \rightarrow G$). The function $\text{CellID}_t(p_a)$ where $p_a \in P$ returns the index of a cell a person is in, defined as

$$\text{CellID}_t(p_a) = \left(\left\lfloor \frac{x}{\delta} \right\rfloor, \left\lfloor \frac{y}{\delta} \right\rfloor\right), (x, y) = \text{Loc}_t(p_a)$$

The set $PG$ represents all the cell locations people are located in at a given time. It is assumed that all people in set $P$ can then be mapped to a grid cell in the set $PG$ at time $t \in T$ as follows:

$$PG_t = \{\text{CellID}_t(p) \mid p \in P\}$$

Assume the set $F$ contains locations $l \in L$ that are of interest to be monitored. We define the set $FR$ as the locations of interest and their surrounding regions as follows

$$FR = \{r \mid l \in F, r \in R(l)\}$$

$PG_t$ can be redefined using the additional condition $Loc_t(p) \in FR$ where $p \in P$ to filter out locations that do not need to be monitored as follows

$$PG_t = \{\text{CellID}_t(p) \mid Loc_t(p) \in FR, p \in P\}$$

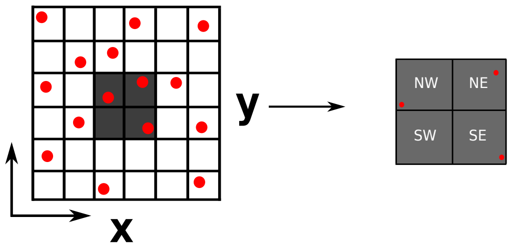



Figure 5.1: Structure of how Points are Stored in Grid Cell

| | T | TL | TR | B | BL | BR | L | R |
|---|---|---|---|---|---|---|---|---|
| NW | SW,SE | SE | - | - | - | - | NE,SE | - |
| SW | - | - | - | NE,NW | NE | - | NE,SE | - |
| NE | SW,SE | - | SW | - | - | - | - | NW,SW |
| SE | - | - | - | NE,NW | - | NW | - | NW,SW |

Table 5.1: Checking Adjacent Cell Quadrants of Cells 1-Adjacent Away From Source

The property of a grid cell implies all people in the same grid cell are within proximity to each other. However there are cases where people can be in proximity that are not in the same grid cell.

Consider the location points in Figure 5.2, it can be seen that there are people that are in proximity while residing in different cells. To efficiently check adjacent cells, we divide each cell into four quadrants. The quadrant cells act as virtual boundaries that can be grouped together efficiently to check for people in adjacent cells that are within $\epsilon$ distance. While our main approach only requires to check active cells and those surrounding, subdividing a cell into quadrants can be used to save checking for extra combinations of encounters. This is more efficient, in particular in areas of greater density.

To efficiently check inside cells, we can create an index inside each cell for some really dense city streets. For the sake of simplicity, we divide each cell into four quadrants in this chapter as an example. Since our main approach only requires to check active cells. Subdividing a cell into quadrants allows for efficient checking of surrounding active cells that are part of the space.

Each quadrant within the cell has a side of $\epsilon/2\sqrt{2}$. In a manner the same as a QuadTree, quadrant assignment is labeled based on the cardinal directions (NE,NW,SE,SW). For example, in the case where a person is in the top left corner of the cell, this would be labeled as NE.

We use virtual boundaries made up of quadrant cells to ensure all people within $\epsilon$ distance are considered. Consider a person $p_a \in P$ in the NE corner of one cell, a person $p_b$ in the SW corner of the adjacent top right cell would satisfy the property $\text{Dist}(\text{Loc}(p_a), \text{Loc}(p_b)) \leq \epsilon$.

The function $\text{CloseQuadsP}_t(c)$ where $c \in PG_t$ returns the set of people in surrounding cells by checking quadrants that are within $\epsilon$ distance of one another. Hence, the people in proximity to the person $p_a \in P$ can be determined by finding all the people within the same cell as $p_a \in P$ and within surrounding quadrants within $\epsilon$ distance.

Initially each person is assigned to a grid cell. At each interval of $\tau$ a scan is performed to determine if a person is still in the same grid cell or has changed cell locations. With $\tau$ being set, only the cells where one or more individuals have stayed longer than a certain amount of time needs to be checked. Such cells are marked as active cells. These cells can be determined by taking the intersection the cells identified at different times, this results in the TimeGrid $TG$ as follows:

$$TG = PG_{tnow} \cap PG_{tnow-\tau}$$

Proximity results are in the TimeGrid output $TG$. A cell in the proximity result must satisfy one of the two following conditions. First, the cell contains at least one person. Second, a surrounding cell contains at least one or more people. The spatial index is refreshed at increments of $\tau$ seconds.

**Algorithm 1:** Building a Spatial Index using TimeGrid

**Input** : A set of people $P$, A timestamp $t$
**Output**: A set of grid cells $TG$

$TG \leftarrow \emptyset$
**for** *each* $p \in P$ **do**
  $c_{now} \leftarrow \text{CellID}_t(p)$
  $q \leftarrow \text{QC}_t(c_{now}, p)$
  **if** $p.c_{prev} \equiv c_{now}$ **then**
    **if** $|c_{now} \cup \mathit{CloseQuadsP}_t(c_{now}, q)| \geq 2$ **then**
      $TG \leftarrow TG \cup c_{now}$
  $p.c_{prev} \leftarrow c_{now}$
**return** $TG$

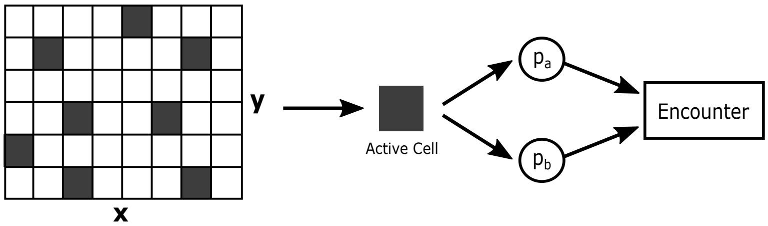



Figure 5.2: Spatial Index using TimeGrid

### 5.4.2 Detecting Encounters

The TimeGrid spatial index allows for the detection of encounter patterns in an efficient manner. Initiating a query to mine for potential encounters $E_t$ can be satisfied by looking up the TimeGrid index. People reside inside cells based on their location. A cell is defined as active if there are more than two people present in the cell or a person can be found in an adjacent cell within $\epsilon$ distance.

To mine for potential encounters, active cells are required to be checked. People within the active cells are within proximity to one another. It is also possible that a person in an active cell is in proximity to people in another active adjacent cell. We can determine who is inside the cell. The function CellP$_t(c)$ where $c \in TG$ returns the list of people inside a cell is as follows:

$$\text{CellP}_t(c) = \{p \mid \text{CellID}_t(p) \equiv c,\ p \in P\}$$

Suppose a person $p_a \in P$ is indexed in a cell $c \in TG$ and no other person can be found in the adjacent cells, then only the people inside $c$ would be in proximity to $p_a$. Thus, the following relationship can be satisfied as follows:

$$\text{c-NN}_t(p_a) = \{p \in \text{CellP}_t(c) \text{ — } c = \text{CellID}_t(p_a)\}$$

Consider CellP$_t(c)$ to be the set of people inside an active cell and CloseQuadsP$_t(c)$ where $c =$ CellID$_t(p_a)$ and $p_b \in$ CloseQuadsP$_t(c)$ to be the set of people in surrounding quadrants that are within $\epsilon$ distance. With these defined, $c$-NN$_t(p_a)$ where $p_a \in P$ in a cell

$c \in TG$ can be defined using TimeGrid as follows:

$$\text{c-NN}_t(p_a) = \left\{ \begin{array}{l} \text{CellP}_t(c) \;\cup\; p_b \text{ — } \\ \qquad\qquad c = \text{CellID}(p_a), \\ \qquad\qquad p_b \in \text{CloseQuadsP}_t(c), \\ \qquad\qquad \text{Dist}(\text{Loc}(p_a), \text{Loc}(p_b)) \leq \epsilon \end{array} \right\}$$

The distance between people found in quadrants needs to be verified to ensure they are within $\epsilon$ distance as points can be on the outer edge of a quadrant that slightly exceed $\epsilon$. The set of encounters $E_t$ can be derived using TimeGrid is as follows:

$$E_t = \{\mathcal{P}(\text{c-NN}_t(p)) \mid p \in P\}$$

The powerset ($\mathcal{P}$) of all the people in proximity is derived resulting in every possible combination of encounters.

Algorithm 2 describes procedurally how to mine for encounters using the TimeGrid index. When a sample needs to be captured, the index is checked for active cells. If the cell is active then adjacent cells will also be checked. The powerset of people all the people in proximity is derived. The algorithm is run multiple times at a sample rate dependent on the level of granularity required.

**Algorithm 2:** Mining for Encounters

**Input** : A set of people $P$,
A TimeGrid spatial index $TG$,
A timestamp $t$

**Output**: A set of encounters $E$

**for** *each* $p \in P$ **do**
  c $\leftarrow CellID_t(p)$
  **if** $c \in TG$ **then**
    q $\leftarrow QC_t(c, p)$
    $E \leftarrow \mathcal{P}(\text{CellP}_t(c) \cup \text{CloseQuadsP}_t(c, q))$

**return** $E$

### 5.4.3 Complexity Analysis

Our approach consists of building the TimeGrid index and the querying for encounters by locating those in proximity using the index. Let $n$ represent the number of people in the search space.

Building the TimeGrid spatial index is bounded by the amount of people in the search space. In Algorithm 1, there is a single loop for iterating over each person to assign a grid cell. Considering there is only assignment and comparison operations, this leads to a time complexity of $\mathcal{O}(n)$, thus allowing for the index to be updated continuously in an efficient manner.

In practise it is very unlikely for our algorithm to yield the worst case. One theoretically possible scenario is that every single person is located in the same grid cell, this would lead to $\mathcal{O}(\binom{n}{2})$. Values typically selected for $\epsilon$ are small so in practise we would not have more than a few people in the same cell.

At query time, encounters are only checked if the person is located in an active cell. Let $m$ to be the amount of active cells and $a$ to be the average number of people inside an active cell. Each person is scanned checking who is in proximity by determining who is in the active cell and the surrounding cells. We assume that people are distributed uniformly at random in an area. This would lead to an average case of $\Theta(m\binom{a}{2})$. Considering only a few people can fit within $\epsilon$ distance, the value of $a$ will be small. We found in our evaluation even for values of $n$ being over a million, only $3\%$ of cells were active. Thus, for small values of $\epsilon$ the average case is closer to $\Theta(m)$.

## 5.5 Encounter Event Simulation

The event simulator is designed to cause encounters to occur for a specified number of people within a city area in a manner it would occur in a real world scenario. Individuals tend to move along a predetermined path when traveling to a point which would likely be not provided to a LBS provider. The simulator uses data from OpenStreetMap which has been demonstrated to be accurate [240] to create paths for people to travel towards.

The simulation continuously iterates over each person and either assigns a destination for them to move towards or leaves them in their current position depending on a probability value. Another person may be selected to go to same destination to cause an encounter to

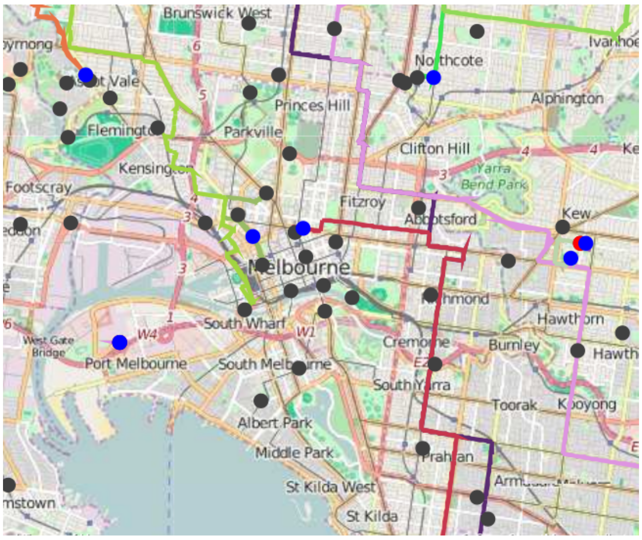



Figure 5.3: Simulation of Encounters in Melbourne, Australia

occur also based on a probability. Each person is assigned a process that handles the moving of objects and finding the shortest path to assigned points. Using OSM data, the location is snapped to the closest node on the road network to ensure the people can travel to the destinations.

The number of people in the simulation determines how many people are traveling and can lead to an increase in encounters in a given area. To ensure encounters occur more often, when organizing a scheduled encounter, only people in the surrounding area are selected. This increases the chances that people would meet people in their local area more often. The duration of an encounter can be set to be between a minimum and a maximum time. The $\epsilon$ value is the minimum threshold considered when people are in proximity. This is usually set to the error rate of the measuring instrument being simulated. For example for GPS it would be $5m$. The speed that users travel can also be set, a suitable value is typically the average speed users travel by foot.

Parameters accepted by the simulator are summarized in Table 5.3. The simulator was used to create data that was used in our experimental evaluation described in the next section.

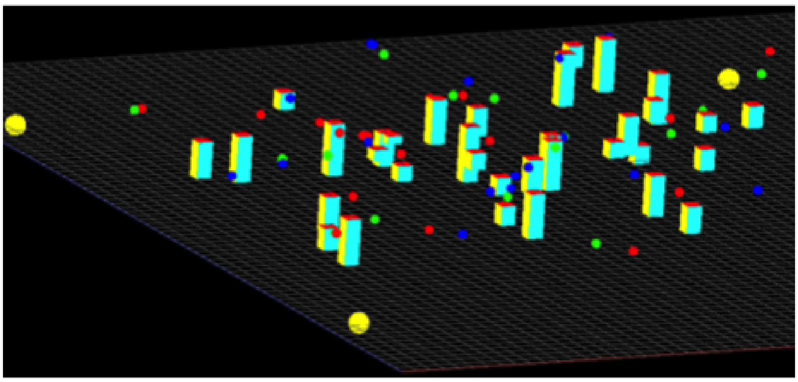


Figure 5.4: TimeGrid Representation in 3D

| Parameter | Description |
|---|---|
| p | Number of people in the search space |
| s | The speed a person is traveling |
| $\epsilon$ | Proximity distance threshold |
| $\tau$ | Update frequency of scan for TimeGrid |

Table 5.2: Encounter Event Simulator Parameters

## 5.6 Experimental Evaluation

This section presents our evaluation comparing TimeGrid with different adapted approaches that make use of spatial indexes for the purpose of detecting encounters. We also test TimeGrid across a variety of typical use cases. Positioning and movement data of people was generated by the simulator described in the previous section and passed to the mining algorithms in real-time.

People were distributed using both a zipfian distribution and uniform random distribution. The zipfian distribution closely mimics how people would be placed in a city, more people would be distributed in areas of higher road network concentration while in uniform random, people would be scattered everywhere within the bounding box with no bias from the road network. People either stay in the one position or move to a destination depending on what the simulator allocates to the person.

The number of people in the city was gradually increased to test the effect of processing large numbers of people that would be typical in a city area. Variations of people were increased in the following order $1,000$, $10,000$, $50,000$, $100,000$, $250,000$, $500,000$, $750,000$, $1,000,000$ and $2,000,000$. Considering modern cities have millions of people, being able to mine in cases in the millions in real-time was the primary focus of our evaluation.

The city we selected to simulate encounters was Melbourne, Australia. We believe it is suitable because the density is high in the inner city areas and is low in surrounding areas. We tested our simulation using an area of $15km^2$, $50km^2$ and $100km^2$ with the center point being set as the CBD to measure the effect of increasing the density of people.

We considered encounters at specific locations and encounters at all possible locations. In the later case, every single person is checked to test if an encounter is occurring. For fixed locations, $10,000$ and $50,000$ encounter location points were monitored in the city area. In the infinite case, the location of each person is required to be scanned to test all possible encounter locations.

The speed in which a person is moving can also make a difference in how the approaches perform. By default people are not moving unless they are assigned a destination in which they will then travel at a certain speed. We considered speeds of $5km/h$ which is the average walking speed, $10km/h$ which is close to a jogging speed, $30km/h$ being the average driving speed in the city center and $60km/h$ which is the driving speed on many main roads in the city.

TimeGrid specific parameters $\epsilon$ and $\tau$ were also tested. $\epsilon$ was set to $5m$ which reasonably represents people close by and can be speaking to each other. Furthermore the lower bound error of GPS technology is also $5m$. $\epsilon$ was also considered to be set to $10m$ and $15m$ which represents people that are within range to be speaking to one another but not necessarily having a direct encounter. The $\tau$ values for refreshing the TimeGrid index were tested for values including 1, 5, 10 and 30 seconds.

The only inputs available to the mining algorithms were the location of a person at a given time. This is typical of what would be available by a service collecting location data in real-time.

### 5.6.1 Implementation Details

The experiments were developed using Java 7 and deployed on Windows 7 running on an Intel i7 950 with 30GB of RAM. Static indexes were built using a QuadTree provided by varunpant[1] and the R-Tree provided by JSI[2]. These libraries provide NN functionality. In

[1]QuadTree: `https://github.com/varunpant/Quadtree`
[2]JSI R-Tree Library: `http://jsi.sourceforge.net`

order to perform a fair comparison, instead of performing a direct NN query, we performed a range query of $\epsilon$ distance which would perform faster since it will only retrieve neighbors that are close.

The static indexes were rebuilt right before a sample was captured. The query region was set to $\epsilon$ distance from the encounter locations. For the QuadTree and R-Tree, a rectangle was generated $\epsilon$ distance from the center point of the encounter location in each direction. This rectangle acted as the query region when calling the intersection with query region method to return the people located at points that are in range. The TPR Tree was provided by libspatialindex[3] which is a popular C++ library. In order to construct the TPR tree, we created a minimum bounding rectangle at the location point of a person and set the velocity rectangle to move in each direction at the speed the user is traveling at. When performing a range query at a location point, the TPR tree returns if that person could have reached that point.

The $B^x$ library we used was provided by MST[4] and was used in the evaluation of [220]. It requires a starting point and a velocity vector. Since it only accepts a fixed vector and not a window, we generated trajectory vectors in multiple directions.

Geohash was used as an index by bucketing points using hashes with a precision of 8 characters. The hashes were generated using the methods provided by the geohash-java library[5]. This leads to grid cells being a size of approximately $38.2mx19m$. When a NN query is executed, a hash is generated for each person. That value is then used to look up a hashtable with each value being set to be an ArrayList object containing people. To run a NN query, a hash is generated at the location and everyone in the buckets is tested to determine if they are within proximity.

The event simulator makes use of many asynchronous callbacks and runs multiple threads to create and track events. As a result the same encounter occurrences would not always occur in the same order. This is attributable to the execution of threads being determined by the CPU scheduler. To account for this in the results, we ran the experiments multiple times and generated an error range. The error range is presented in the charts as error bars.

[3]LibSpatialIndex: `https://libspatialindex.github.io`
[4]$B^x$ code: `http://web.mst.edu/~lindan/`
[5]Geohash: `https://github.com/kungfoo/geohash-java/`

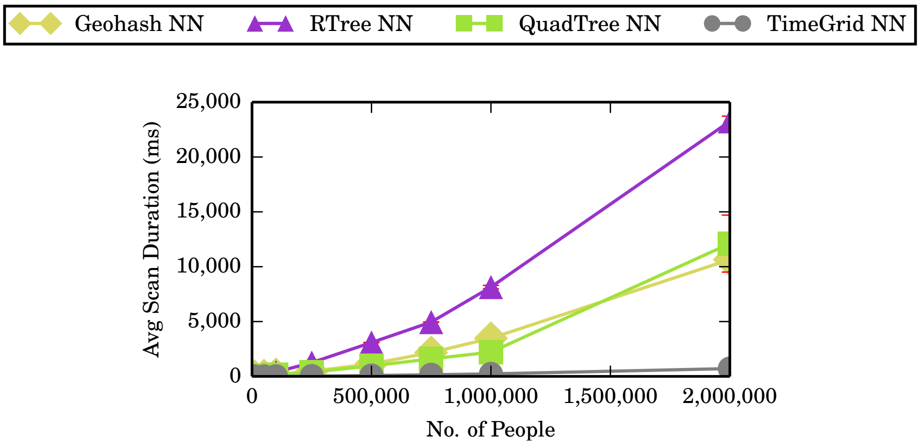



Figure 5.5: Avg Scan Duration by Number of People
$A$: $30km^2$, $\tau$: 1s, $\epsilon$: 5m, $\nu$: $5km/h$

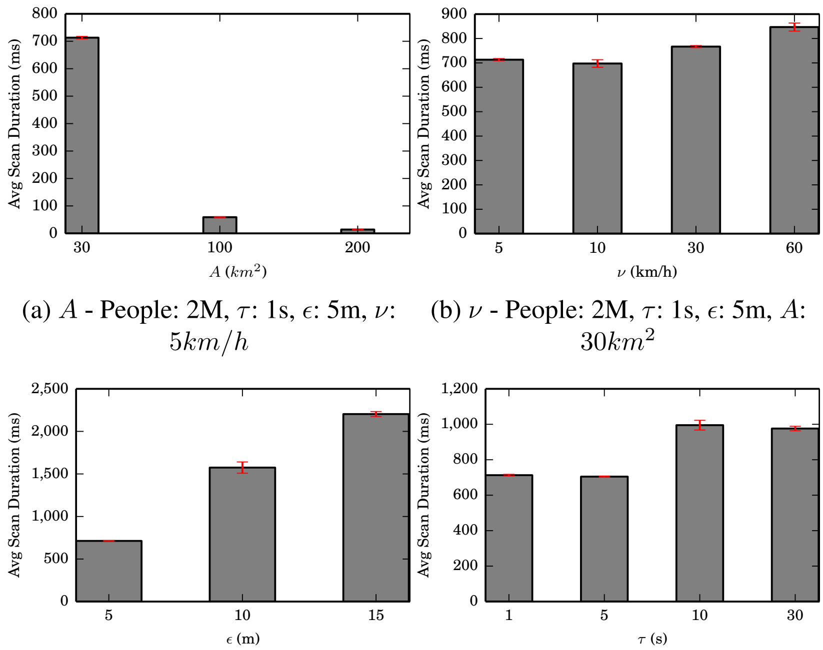
| Parameter | Values Tested |
|---|---|
| # of People | 1k, 10k, 50k, 100k, 250k, 500k, 750k, 1M, 2M |
| $\epsilon$ | $5m$, $10m$, $15m$ |
| $\tau$ | $1s$, $5s$, $10s$, $30s$ |
| Speed $\nu$ | $0km/h$, $5km/h$, $10km/h$, $30km/h$, $60km/h$ |
| Area $A$ | $30km^2$, $100km^2$, $200km^2$ |

Table 5.3: Parameters Tested in Evaluating Algorithms to Mine for Encounters

### 5.6.2 Results and Discussion

TimeGrid demonstrated that it significantly outperforms competing approaches in every use case we tested. It can scan for encounters in under a second even in situations where millions of people were considered and every possible encounter location was tested. We first compare TimeGrid against certain nearest neighbor algorithms that use static indexes [224, 225] as can be seen in Figure 5.5.

Mining for encounters can be done using indexing of moving objects. We found that velocity based approaches did not work well for our problem at all. For example, when indexing moving objects with TPR tree, we found that scan duration grows substantially as the time period of the simulation increases. Figure 5.7 plots the time elapsed since the TPR tree was built with the scan duration. It can be seen that the scan duration grows significantly as time increases. The trend resulted from the fact that when a person moves, the amount of places the person could possibly visit increases as time elapses. TPR trees tend to perform better for queries where the the common velocity vector is fixed in a limited direction. For example, which airplanes are going to land on the runway in the next ten minutes. We also




(a) $A$ - People: 2M, $\tau$: 1s, $\epsilon$: 5m, $\nu$: $5km/h$

(b) $\nu$ - People: 2M, $\tau$: 1s, $\epsilon$: 5m, $A$: $30km^2$

(c) $\epsilon$ - People: 2M, $\tau$: 1s, $A$: $30km^2$, $\nu$: $5km/h$

(d) $\tau$ - People: 2M, $\epsilon$: 5m, $A$: $30km^2$, $\nu$: $5km/h$

Figure 5.6: Varying Parameters of the TimeGrid Algorithm

attempted a $B^x$ tree, however we found that this approach cannot be applied to our problem due to the requirement of fixed velocity vectors.

A continuous approach to answering $k$-NN queries for moving objects proposed in [232] was also tested. In attempting to apply this method, the evaluation time took much longer than any other method we used mainly due to cases where the closest neighbor is far away. When looking for $k$ neighbors, grid cells would incrementally expand until it finds at least one neighbor. It would take significant time to find even just one neighbor in cases where the closest neighbor is far away. In our experiments, this situation occurred many times. In one preliminary test, when considering just a thousand people with 50 fixed encounter locations, this approach took an average of 14 seconds to find encounters. As more people and encounter locations were considered, the approach could not scale.

Increasing the size of the search space has the effect of lowering how densely packed people are within the tested area. It can be clearly seen in Figure 5.6a that increasing the search space improves the scan performance as people will be more sparsely distributed.

Varying the speed did not appear to make a major difference in performance as seen in Figure 5.6b. While traveling from one grid cell to another quicker may reduce how many

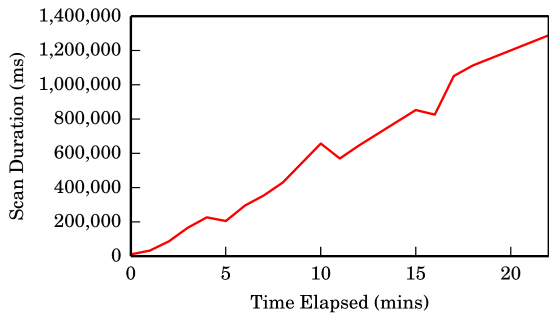



Figure 5.7: TPR Tree Performance as Time Increases

active cells are indexed, there is a large number of people in the city that do not move and stay in the same position for long periods of time.

In theory, increasing the size of $\epsilon$ causes the TimeGrid to take longer to scan for encounters, this is due to having to check for more combinations in the powerset. This is evident in Figure 5.6c. Larger values of $\epsilon$ would result in people being too far away to have an encounter.

Increasing the interval $\tau$ reduces the amount of times the index needs to be updated, however possible encounters could be missed. Changing how often the index was updated had no real noticeable trend for affecting the scan time. This is shown in Figure 5.6d.

Initially, people are placed at a random location and move at the assigned speed of $\nu km/h$. Our aim was to realistically simulate the movements in a city environment. We also tested an extreme case of 2 million people not moving with the parameters set as $\nu = 0km/h$, $A = 30km^2$, $\tau = 1s$ and $\epsilon = 5m$. TimeGrid ran in just over a second on average even though movements over time were not being exploited. Thus, TimeGrid still performs well whether people are moving or not.

Our evaluation demonstrated how TimeGrid is better suited for mining encounters. While other techniques may miss quick encounters, our approach would capture them. We also gave insights into why current velocity based techniques designed to handle moving objects are not suitable for our application domain. Overall, the TimeGrid algorithm performed orders of magnitude better than conventional techniques across all the use cases we tested. Samples were captured fast enough to detect encounters in real-time, even if they last for just a second. We demonstrated that our approach is capable of mining encounters in real-time for a city of two million people.

## 5.7 Conclusions and Future Work

This chapter shows that once people's location is deducted, encounters between individuals which is seemingly undetectable can be detected in real-time. This is achieved by efficiently indexing people within proximity. Encounter information not only provides context of what a user may be doing based on their location but also the interactions that are occurring with individuals in their surroundings.

In this work, a new class of data mining problem has been defined, detecting encounters using spatial data. Data mining in very large data sets in real-time has received much attention recently as conventional data warehouses are now capable of processing on the fly.

A novel technique was also presented that mines for encounters in an efficient manner. This technique is based on the spatial properties that for an encounter to occur, users need to be in proximity to each other for a certain amount of time. These properties were used to build a new spatial index that can be easily looked up to determine encounters quickly.

An advanced simulator to create encounter occurrences using real city data was presented simulating the people moving in the city and organizing for encounters to occur. The simulator makes it possible to test a range of different techniques to mine for encounters in many different scenarios.

Our evaluation demonstrated how our algorithm is better suited for mining encounters. While other techniques may miss quick encounters, our approach would capture them. We also gave insights into why current velocity based techniques designed to handle moving objects are not suitable for our application domain.

For future work we plan on expanding our approach to consider moving encounters that occur across multiple locations after they begin due to individuals moving. Another possible direction is to deploy our algorithm using a real social network data and running a user study.

# Chapter 6

# Identifying Smartphone Users Using Only Diagnostic Features

.

In the previous chapters we explored how query results could reveal a smartphone user's location. Mobile smart phones capture a great amount of information about a user across a variety of different data domains. This information can be sensitive and allow for identifying a user profile, thus causing potential threats to a user's privacy. In addition to query data, smartphone apps send diagnostic data to service providers for the purposes of debugging and improving the quality of service.

Many mobile games and applications collect diagnostic data as a means of identifying or resolving issues. Diagnostic data is commonly accepted as less sensitive information. There are notable open source apps that share diagnostic data online in bug tracking systems. Types of data include the type of phone and how much storage space is available. By looking at one diagnostic metric in isolation to others would not reveal much. For example, hundreds of millions of people have a device with the manufacturer metric specified as *"Samsung"*. We consider both static and dynamic diagnostic features. Static features such as the manufacturer do not change on a particular device while dynamic fields such as storage space change over time. Combining a few fields leads to the number of possible smartphone users that could result from certain diagnostic data to drop off rapidly.

This chapter demonstrates how diagnostic information that is not considered sensitive, could be used to identify a user after just three consecutive days of monitoring. We have used

---

**This chapter is based on**
**Is this you?: Identifying a Mobile User Using Only Diagnostic Features - MUM 2014**

the *Device Analyzer* dataset to determine what features of a mobile device are important in identifying a user. This dataset has gained popularity amongst mobile privacy researchers as it has captured data for a large number of users. Users can set the types of data they wish to share. While many metrics like location were switched off, almost every user provided diagnostic data.

Our experimental results demonstrate that using only diagnostic features like hardware statistics and system settings, a user's device can be identified at an accuracy of 94% with a Naive Bayes classifier. Diagnostic data is considered to not be sensitive and exchanged freely. By demonstrating that it can uniquely reveal a user, providers need to take care in how it is shared.

## 6.1 Introduction

Modern smart phones capture a great amount of personal information about a user across a variety of different data domains. Extractable diagnostic features could be collected on a regular basis by a large number of mobile apps by the fact that many smart phones have Internet access.

Mobile app marketplaces such as Google Play and Apple App Store are convenient for both the application developers and the mobile users providing centralized services for downloading third party applications. This has led to an explosion of mobile application development and their usage [241]. Many of these applications capture raw data from a user's device and upload it to a remote database in order to deliver certain services. While these applications can provide significant benefit to the users, they can also impose potential risk to disclose sensitive user information.

In smart phone applications, location data collected via GPS, Wi-Fi, RFID or Bluetooth sensors is considered as the most sensitive information causing the most severe privacy risks [242, 243, 244, 245]. Sensitive personal data can also be captured through camera, microphone, accelerometer sensors installed in smart phones. There is also other seemingly less-sensitive information such as signal strength information, hardware statistics or system settings that could be easily accessed. Signal strength information is considered to be diagnostic because it is often used to determine strength of wireless connections and if devices are in range. These features can be extracted by a mobile application like Device Analyzer[1].

In this chapter, we have analyzed the Device Analyzer dataset [246] to see what features are important in order to identify a user's device other than the obvious sensitive information. To make the data more accessible for our analysis, we first transformed the raw dataset to a aggregated dataset to provide context about each user at a daily level. A web application has been developed as a part of this aggregation process to describe the daily level context of a Device Analyzer user.

We have modeled a Naive Bayes classifier to learn a user's device using less sensitive features such as hardware statistics. Our experiment shows that using only information like manufacturer name, internal and external memory usage and system settings, a user profile can be predicted at an accuracy of 94%. Only three consecutive days of monitoring diagnostic features are necessary to identify a user profile, a time period that is short for normal app usage. We also collected Bluetooth signal strength information of surrounding devices and demonstrate how it could be used to infer contextual information.

A mobile app has access to direct features that can uniquely identify a device such as a WI-FI MAC address, however diagnostic information used in our experiments is less suspected in posing as a private threat and more widely distributed to remote servers. For example, a mobile hardware manufacturer company may have access to the usage of hardware statistics of its customers for analyzing the performance.

This finding is a threat to user privacy as an adversary could learn the identity of a user profile given they have access to an additional dataset that contains the user's name or if a user moves to pay for a service and reveals their name to complete a transaction. Once the identity of a user is known, this could lead to further intrusions. For example, a user may be targeted with unsolicited marketing.

Our main contributions are as follows:

- We provide an approach to aggregate the dataset at a daily level.
- We developed a web application called DescribeMyData to describe the context of a Device Analyzer user at daily level in human readable format.
- We demonstrate that diagnostic features of a mobile device such as hardware statistics

[1]Device Analyzer App: `https://play.google.com/store/apps/details?id=uk.ac.cam.deviceanalyzer`

and system settings can be sufficient to predict the user.

- Using a Naive Bayes classifier, we obtain a accuracy of 94% to predict a user's device from less sensitive mobile information.

## 6.2 Background

Smart phones are becoming the number one convergence device that stores most of a user's personal data. Information privacy in pervasive computing is a established research discipline with roots dating back to the early 1980s when PCs were gaining mainstream adoption. There is now growing concern of the privacy implications associated with smart phones.

Traditionally smart phones have been used for location based services and social networking applications. As a result considerable research has been performed to protect user privacy in location sharing applications [242, 243, 244, 245]. For example, Consolvo et al. have discovered the three important factors, why, what and when for sharing location information with service providers [242]. In [244], a privacy-aware location sharing application, called *Locaccino*, has been developed. Anthony et al. have studied whether users' preferences to share location information vary with respect to place or social context [243].

Research into the privacy implications of third-party app usage is a relatively new topic. Automated systems have been proposed for both Android and iOS to detect privacy leaks at an API method call level. While these systems are very useful in detecting where sensitive information is leaked, they do not indicate if it is justified given the functionality of the app.

With the advancement of smart phone applications, people are becoming more concerned about the privacy of other sensitive data in their phones, for example photos, contacts, etc. According to the survey performed by Ben-Asher et al., the sensitivity of mobile data depends on the data type and the context of use [247]. Chin et al. have performed a survey on 60 smart phone users to determine their attitudes towards security and privacy. They found that people are more concerned about privacy on the smart phones than their laptops [173].

A real-time privacy monitoring application, called *TaintDroid* was recently developed for smart phones. According to their result, *TaintDroid* could identify misusage of users location and device identification information for 20 applications out of 30 popular Android applications [241].

The iOS platform relies on a strict auditing process to protect their users as opposed to a permissions system. A system called PiOS [170] gained much attention demonstrating the ability to deconstruct an iOS application and demonstrate where privacy leaks occur. PiOS found that most of the applications that were analyzed on iOS do not leak much personal information, however more than half leaked the device ID.

Another system called Stowaway was demonstrated in [9] which compares method calls made by an app relative to the permissions the app developers request in the Android Manifest. Interestingly it was found that one third of applications are not following the least privilege path with their permission requests. For example, a developer may request fine grained location access but only actually use coarse grained access. This is likely a result of developers not understanding how to request permissions correctly due to unclear documentation.

Similar work has been performed in the online privacy area by using browser configuration features to determine if a browser can be uniquely identified in [248]. It was demonstrated that given common plugins like Java and Flash are installed, 94.2% of browsers in the sample are unique.

In this work, we argue that less-sensitive mobile information such as hardware statistics and system settings can cause potential threats to a user's privacy.

## 6.3 Problem Definition

Consider the set of users $U$ that use smartphones that capture diagnostic data. Each user $u$ in a set of users $U$ communicates a set of smartphone diagnostic features $F$ at a time $t$. The time $t$ is usually when diagnostic data is sent for analysis for the purposes to improve the quality of an app. The aim is to build a profile about a user $u$ using a known training dataset containing diagnostic features to identify a user in another dataset where the identities of the user are not known.

## 6.4 Methodology and Experiments

In this section, we describe our approach for identifying users based of diagnostic data and evaluate it using data from the Device Analyzer dataset [246]. We also present preliminary findings of how the signal strength information provided by Bluetooth enabled device can be used to infer contextual information of user behaviors.

### 6.4.1 Naive Bayes for Data Modeling and Classification

To uniquely identify a user, the model needed to classify the handset ID as it identifies the user's device. Naive Bayes which is based on Bayes' theorem that captures the probability of an event occurring based on possible conditions that effect the event. It was a good candidate given for our problem given it has proven to be efficient when there are many classes [249].

We take Naive Bayes and formulate it for our problem. Consider the set of user IDs $U = \{UID_1, UID_2, ...\}$ to be the classes that need to be classified and $F = (f_1, ..., f_n)$ to be a set of features where $n$ is a the number of independent variables. Naive Bayes makes a strong assumption of independence between features.

$$
\begin{aligned}
U_{MAP} &= \underset{u \in U}{\operatorname{argmax}} P(u \mid F) && \text{MAP is the Maximum Posteriori} \\
&= \underset{u \in U}{\operatorname{argmax}} \frac{P(u \mid F)P(u)}{P(F)} && \text{Apply Bayes' Theorem} \\
&= \underset{u \in U}{\operatorname{argmax}} P(u \mid F)P(u) && \text{Remove Denominator} \\
&= \underset{u \in U}{\operatorname{argmax}} P(f_1, ..., f_n \mid u)P(u)
\end{aligned}
$$

Conditional independence assumes the feature probabilities $P(f_i \mid u_j)$ are independent given the user ID $u$. This assumption allows for a classifier to be derived using the equation below:

$$
U_{NB} = \underset{u \in U}{\operatorname{argmax}} P(u_j) \prod_{f \in F} P(f \mid u)
$$

### 6.4.2 Dataset

Given the nature and scale of the data, we followed a traditional data-ming approach performing preprocessing on the data, feature selection and modeling. A web application was developed to view Device Analyzer user data. We have used the Device Analyzer dataset [246] to see what features are important in order to identify a user's device other than the obvious sensitive information. The complete Device Analyzer dataset contains smart phone usage from over 17,000 devices. Given the nature and scale of the data, we

| Feature | % Off | % On | % Null |
|---|---|---|---|
| System Settings Lock | 52 | 24 | 24 |
| System Settings Sound Effects | 50 | 36 | 14 |
| System Settings Device Stay On | 85 | 9 | 7 |

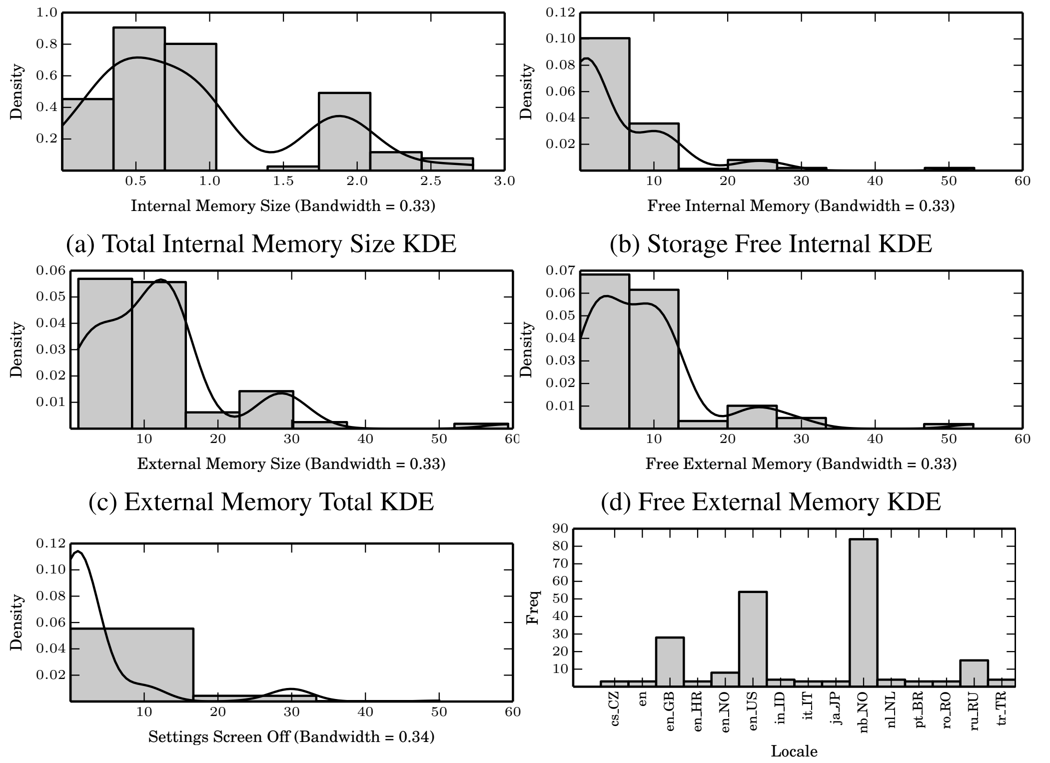
Table 6.1: System Settings Features Distribution

followed a traditional data-mining approach performing preprocessing on the data, feature selection and modeling. A web application was developed to view Device Analyzer user data. Our experimentation focused on finding correlations between features and establishing their predictive power.

### 6.4.3 Preprocessing

The dataset generated from the Device Analyzer app is stored in a low-level detailed format. To make the data more manageable, we parsed the data extracting all key and value pairs and aggregated it to a daily level per handset. User profiles that did not capture memory usage were filtered out of our model. Preprocessing data in daily format is suitable because these fields would not change frequently and the size reduction of the dataset makes it feasible to train a classifier. Looking at data in daily format also allows trends to be easily identified. The following process was performed:

1. Iterate over every data file for each Device Analyzer user;
2. Extract handset ID and date keys and create a data structure to store daily level data;
3. Use the handset ID and date as a hash to store each daily data structure in a hash table;
4. When a numerical feature is found, derive SUM, COUNT, AVG, MIN, MAX and update the daily data structure;
5. When a categorical feature is found e.g. system settings lock mode, take the value at end of the day and update the daily data structure;
6. Produce an output file for each device after all data is iterated;
7. Combine each output files into a single file which can be uploaded into a relational database.




(a) Total Internal Memory Size KDE

(b) Storage Free Internal KDE

(c) External Memory Total KDE

(d) Free External Memory KDE

(e) Screen Off Minutes KDE

(f) Locale Distribution

Figure 6.1: Kernel Density Estimation (KDE) Plots of Continuous Features

### 6.4.4 Feature Selection

Our aim was to test the predictive power of diagnostic features, the following features were used for our model:

- Device Manufacturer
- Device Model
- Device Locale (Language Setting)
- Internal Memory Size
- Free Internal Memory
- External Memory Size
- Free External Memory
- System Settings Lock
- System Settings Sound Effects Toggle
- System Settings Screen Stay On Duration
- System Settings Device On While Charging

The dataset indicates that a wide variety of Android devices are being used. After pre-processing, our dataset contains 18 mobile manufacturers and 56 mobile device models.

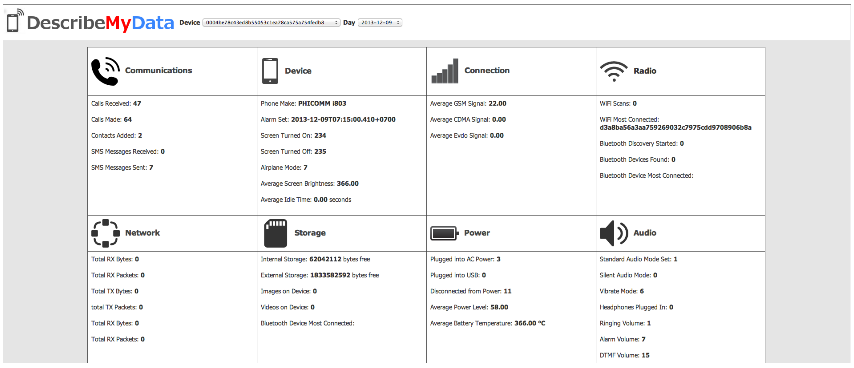



Figure 6.2: DescribeMyData: Web Application to Visualize Daily Level Data of Device Analyzer Users

To estimate and visualize the distribution of the continuous features across user profiles, Kernel Density Estimation (KDE) was preformed. In Figure 6.1a, it can be seen that total internal device memory is more evenly distributed than internal free memory in Figure 6.1b. Conversely, in Figure 6.1c and Figure 6.1d, both external memory available and external memory free is well distributed. Most user profiles appear to have a similar setting for how long the screen stays on when there is no user input as described in Figure 6.1e. There is also a bias in the dataset to specific locales as shown in Figure 6.1f.

Percentage distributions of discrete features is presented in Table 6.1. Half the population of users do not specify a phone lock setting on a given day while a third of users have a lock. Sound effects are turned off be at least half the users in the population and most users allow the device to turn off while charging.

### 6.4.5 Implementation

The parser was developed in C using libraries *libcsv* and *uthash*. After preprocessing the data, the output file was imported into a MySQL[2] relational database to allow for analysis and feature extraction. Feature selection was performed in Weka 3[3] using the InfoGainAttributeEval method. R[4] was used for modeling making use of the e1071 library.

To easily visualize daily level data for each user, we also developed a web application called DescribeMyData as shown in Figure 6.2.

[2]MySQL: `http://www.mysql.com`
[3]WeKa: `http://http://www.cs.waikato.ac.nz/ml/weka/`
[4]R Project: `https://www.r-project.org/`

| Train/Test Split (%/%) | Accuracy (%) | Macro Avg Precision | Macro Avg Recall |
|---|---|---|---|
| 70/30 | 93.75 | 0.921 | 0.949 |

Table 6.2: Experimental Results Using a Naive Bayes Classifier

### 6.4.6 Experimental Results

Using only the diagnostic features described in *Feature Selection*, the model produced using Naive Bayes was accurate.

Our sample for analysis contained 223 days of data in which 66 user profiles could be uniquely identified. In the preliminary analysis we trained and tested our model on devices that contained at least two days of captured data, however we found that this was not sufficient to uniquely determine a user. Thus, only devices in the dataset with at least three days of captured data were considered for analysis. In practice three days is still a small amount of time because a mobile user will use apps for significantly longer periods which are likely to increase the accuracy of our approach further. Days in which the Device Analyzer app did not capture the external memory free feature for a device were also filtered. Numerical memory features were scaled based on the maximum for the respective feature.

The dataset was distributed ensuring that each device has 70% of the data points as training data and evaluated on the remaining 30%. Table 6.2 describes the accuracy, precision and recall values of the classifier. It can be seen that a user's device can be identified at an accuracy of 93.75% with our model. The macro average of precision and recall values across all the classes are 92.1% and 94.9%, respectively.

### 6.4.7 Invasive Bluetooth Beacons

Bluetooth devices emit a radio signal that can be perceived by other Bluetooth enabled devices that are in range. Over a three week period, we ran a preliminary experiment using only the Bluetooth device discovery features. The mobile was left on a desk inside a University lab and was not moved. Bluetooth discovery data can provide the time the sensing device can sense another device and signal strength to approximate distance.

Considering that we found many Bluetooth users had their device name set to something that is personally identifying such as their full name, we anonymized the dataset by removing this information. Figure 6.3 displays the results with the y-axis representing a unique device and the x-axis representing the sample time of when the device was captured. Interestingly, we were able to determine the times staff were in their offices and even the times in which some desktop PCs were active as the Bluetooth receiver activates when the machine is not on standby.

The user is not aware when signal information emitted by their device is captured by nearby devices. This type of information can be considered diagnostic but reveals general habits when collected over a period of time.


Figure 6.3: Sensing Bluetooth Devices in Range in an Office Space

## 6.5 Conclusions and Future Work

In this chapter, we presented a methodology to predicting a device ID based on diagnostic features. We also presented a web application that can be used to easily navigate through the Device Analyzer dataset and present the information in a more comprehensive visual format.

Our results indicate that when using features that would not commonly be considered sensitive are captured over a number of days, a simple Naive Bayes classifier can produce an accurate model to identify a user.

For future work, a possible direction is to determine if certain features can be used to predict other features in the dataset.

# Chapter 7

# Smartphone Privacy Protection with PrivacyPalisade

Throughout this thesis we have revealed how even seemingly innocent data can be used to infer sensitive information. Privacy has become a key concern for smartphone users as many apps have the ability to access and share sensitive data. However, it is not easily understandable for users which apps access what type of data if an app is only using the minimal access permissions that are required to achieve a certain functionality. Smartphone users that install many apps on their device increases the risk that their data will be aggregated on a central data warehouse server where sensitive inferences can be made. In this chapter, we propose a solution to help protect smartphone users.

Although there are apps targeting such privacy concerns, they only show which type of data is being accessed but not whether it is necessary for an app to achieve its functionality. We propose a model that groups apps together in terms of advertised functionality and assesses an app's privacy intrusiveness based on the requested permissions relative to other apps that provide similar functionality. If an app requests a permission that is not common in its cohort, the user is notified and shown visually the implications of that permission. The user can then decide on what action to take. To improve user comprehension of permissions, we implemented PrivacyPalisade and demonstrate Android OS level modifications that use visual cues to indicate privacy intrusiveness of an app. Our approach is scalable and incurs little performance overhead to the system.

The privacy implications of using smartphones is becoming well known to the general

---

**This chapter is based on**
**PrivacyPalisade: Evaluating App Permissions and Building Privacy into Smartphones - ICICS 2015**

public. This understanding will lead to demand for privacy aware smartphones. We demonstrated throughout our work how even data that appears to be non-sensitive can be analyzed to reveal sensitive insights. Thus, the protection of privacy for smartphone users is a hard problem to solve. The system we propose aims to detect apps that are requesting permissions simply to mine for data and alert the user. This will allow the user to download apps that are more privacy conscience.

## 7.1 Introduction

The ubiquity of mobile smartphones combined with advancements in mobile network infrastructure has created a strong market for third party apps. While the apps and service providers are of great convenience to its users, there are concerns of the privacy implications [1]. Mobile apps greatly enhance the experience of using smartphones. These apps are typically developed by both large software firms and independent programmers [250]. Modern platforms allow for third-party developers to create apps and distribute them in app stores or marketplaces. Popular app stores for Android include Google Play, Amazon Appstore and GoAPK while iOS users primarily download apps from the Apple AppStore. Google Play and Apple AppStore now list over one million apps.

Third-party developers can access a large amount of user data using standard API calls provided by the mobile platform and send it directly to remote servers. Apps often make use of user data to provide functionality. For example, VoIP apps such as Viber require access to a user's contacts list to provide a list of people the user can call. For this case, iOS displays a popup asking the user if the app is allowed to access contacts. In contrast, Google Play only alerts the user at installation time. The contacts permission is justified for the VoIP app in order to provide auto-dialing. However, many other apps, such as weather apps, do not require the permission in general.

The architecture for privacy protection varies between mobile platforms. Android depends on a software based permissions system. When an Android user downloads an app, a dialog at installation explains what data an app can access. In contrast, Apple employs staff members that manually review apps with internal checks. Both approaches are not without issues. It has been shown that user comprehension of permissions is low among Android users. While iOS users need to place trust in the AppStore review process which is not entirely transparent. A more comprehensible privacy solution is needed that does not unnecessarily hinder app functionality while at the same time protects user privacy.

App similarity has been provided as a measurement by marketplaces. It is commonly based on download statistics across apps. For example, if users downloaded one app, how many of those same users downloaded another app. We propose a protection method that uses app similarity to detect anomalies. From building a table of permissions requested of an app and comparing it with the permissions of those of similar apps, anomalies can be detected. We use Isolation Forest [21], a data-mining technique for detecting outliers to find the anomalies based on the requested permissions.

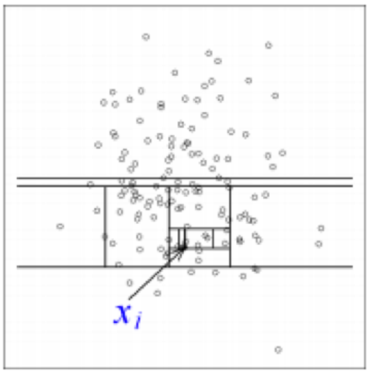
We focus on Android, which is a popular open source Linux based mobile OS designed for smartphones and tablets. Android is used to prototype our approach as its open nature allows changes to be made to the privacy models. As of 2014 Android has an estimated global marketshare of 81.5% in a report conducted by IDC[1], thus privacy research of the platform is of great importance.

To evaluate our approach, we developed a web scraper and collected information of nearly 17,000 of the most popular free and paid apps from the Google Play store. For each app, data collected by the web scraper includes the permissions required by the app and a list of similar apps as suggested by Google Play. We demonstrate our approach for flagging outliers in app permissions will eventually lead users to pick apps that take more conservative paths to user data access.

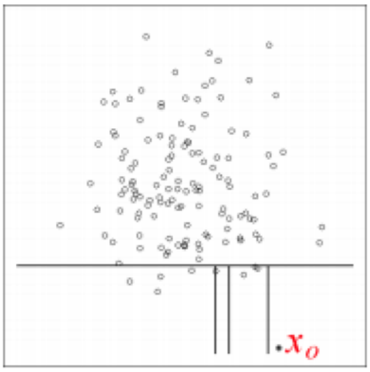
Based on our model, we implemented a service called PrivacyPalisade that checks apps installed on the smartphone. If an app is found to contain permissions that are outliers, the user is notified about the nature of the data the app requested at launch time. We made OS level modifications to achieve this. Our evaluation shows that PrivacyPalisade does not add much overhead to the system and uses little resources. It works across free and paid apps as it does not need access to the bytecode files.

Detecting potentially harmful Android malware using requested app permissions as training vectors for use in data-mining approaches has been attempted before in [251, 252, 253]. PrivacyPalisade is more privacy focused and differs by determining if permissions are justified relative to app functionality. The permissions of similar apps are used as means of comparison to achieve this.

[1]IDC Report: `http://www.idc.com/getdoc.jsp?containerId=prUS25450615`




(a) Isolating Data Point $x_i$




(b) Isolating Data Point $X_o$

Figure 7.1: Detecting Outliers with Isolation Forest [21]

In summary our key contributions are to:

- Propose an approach to highlight outlier permissions relative to an app's category and functionality;
- Implement PrivacyPalisade, a ready-to-deploy application that highlights privacy implications to Android users;
- Demonstrate OS level modifications that receive PrivacyPalisade messages to help alert users of privacy implications;
- Provide case studies of apps that we believe do not follow the path of least privilege.

## 7.2 Background

### 7.2.1 Isolation Forest Overview

Isolation Forest is a unique anomaly detection technique as it builds a profile that explicitly isolates anomalies as opposed to building a profile of normal points and finding those that do not conform [21]. Given our problem, in many cases there are only a small number of similar apps to compare to a target app. Thus, Isolation Forest is a suitable choice because it has demonstrated high performance with minimal training.

The motivation behind the Isolation Forest approach is that anomalies can be isolated easier than other data points in a set. Consider the example illustrated in Figure 7.1 demonstrating this principle. Isolating $x_o$ which is an outlier only requires a few random partitions while isolating $x_i$ requires many more. This idea idea has been shown to hold true in many different datasets [21].

An Isolation Tree is a proper binary tree defined to contain nodes that are either external and have no children or internal with one test and two child nodes. The test is made up of an attribute value and a split value that divides the data. The split value is selected randomly and is between the minimum and maximum range of the attribute selected. A dataset is recursively divided by selecting a random attribute until the tree reaches its set height, there is only one remaining attribute or all remaining data points yet to be selected have the same value. The tree represents the amount of divisions to isolate points that can be then used to rank the degree of anomaly.

The anomaly score is defined to reflect how much of an anomaly a data point is. Given a point $x$, the function $h(x)$ measures the number of edges that need to be traversed in an Isolation Tree. The issue with deriving an anomaly score from $h(x)$ is that the height of the tree can grow depending on the number of entires in the dataset. This makes it difficult to compare directly different values of $h(x)$.

Since the Isolation Tree has the same structure as a Binary Search Tree, $h(x)$ can be estimated by considering the case of an unsuccessful search in the Binary Search Tree. The estimation of the average $h(x)$ given $n$ is defined by the function $c(n)$, this can be used to normalize $h(x)$. The function $c(n)$ is provided in [254] as follows:

$$c(n) = 2H(n-1) - (2(n-1)/n), \text{ where } H(i) \approx ln(i) + 0.5772156649$$

Given a set of Isolation Trees, the average of a set of isolation trees is defined by the function $E(h(x))$. The anomaly score $s$ for a data point $x$ can then be defined as follows:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Scores of $s$ being close to 1 indicate a definite anomaly while scores of $s$ being closer to 0.5 indicate the points are quite safe and likely not an outlier. These scores are by PrivacyPalisade to determine the likelihood an app is requesting outlier permissions.

### 7.2.2 Android Overview

In this section we provide an overview of the Android operating system, permissions used to enforce security and how users download third-party apps via marketplaces to make our work more tangible.

Android is an open source mobile operating system owned by Google. It was originally designed for smartphones and tablets but is now found on a wide range of devices including watches, smart televisions and digital cameras. The OS has been embraced by many device manufacturers including Samsung, LG and HTC.

The Android operating system is built on the Linux kernel. Android applications are typically built in Java on top of the Android API which is a set of libraries. Java SDK class libraries can also be used and third party libraries can be imported. It is possible to also use method calls not documented in the official API by inspecting the Android OS source code. It has been found that apps do in fact use undocumented calls [255]. While applications are compiled using the Java SDK provided by Oracle, Android itself runs a custom Java VM called Dalvik. Apps contained in an Android application package (APK) which is similar to the JAR file format.

Each app defines its components in a Manifest file which provides information to the system. An Android application usually contains a combination of app components including Activities, Services, Content Providers and Broadcast Receivers. Activities are defined as a screen that comes with a user interface while services usually run in the background. Content providers are used to manage data while broadcast receivers send announcements to apps in the system. For example the system can send a broadcast to apps that the screen has been locked or that a new email has been received.

To facilitate communication within applications and between applications, Inter-process communications (IPC) is commonly used. IPC mechanisms introduced by the OS includes Intents, Bundles and Binders. Intents are messages which can be broadcasted to all applications or a specific set. They are commonly used to start activities and services. Bundles are used to pass Objects such as Strings and Bitmaps. An Intent can be associated with a bundle. Binders allow an application to specifically make a method call on another running process.

Each app is run as a separate operating system process and in an isolated JVM, in addition apps are run with their own Linux User and Group ID. Developers can create third-party applications using Android build tools, at present Android Studio is the official tool promoted by Google.

### 7.2.3 Android Permissions System

Android primarily relies on system features engineered into the operating system to protect users from malicious apps. Developers statically declare what permissions they require in a Manifest file. The system blocks API calls that do not have permission by terminating the app. A permission restricts access to potentially sensitive data or code on a device that could cause harm such as making automated calls to premium numbers without consent.

There are four protection levels assigned to permissions which indicate potential risk, *"normal"*, *"dangerous"*, *"signature"* and *"signatureOnSystem"*. Normal permissions are considered low risk, dangerous permissions will prompt the user explicitly and signature requires the application be signed with the same certificate. Permissions with a risk value set to *"signatureOnSystem"* require the application be signed with the same certificate as the system image. In practice most permissions use *"signature"*. Third-party apps downloaded from Google Play are required to be signed.

In Android 7.1.1 Nougat (API Level 25) there are 277 Android system permissions. Each feature is protected by at most one permission. Each Android application contains a manifest formatted in XML, which is called *AndroidManifest.xml*. Android developers place XML code in the manifest defining which permissions the app will be requesting while in execution. Defined in Listing 1 is an example of how a developer would request sensitive permissions.

Third-party applications can also protect components for use as modules in other apps by using the Android permissions system. For example, Google Play is also a library that provides web services. To use the Google Play library, a developer would be required to declare the permission in Listing 2 and defined in Listing 1 is an example of how a developer would request sensitive permissions.

### 7.2.4 Android Marketplaces

Android users can browse and download many apps for their smartphones from a variety of markets. The market ensures apps that are listed are compatible with a user's device. Apps are submitted by developers that wish their app to be downloaded. App marketplaces are responsible for alerting users to the potential privacy implications at installation time. Developers that wish to list their apps in the store upload an APK file and enter all meta-data related to the app such as category and description.

Google play utilizes a simple layout and divides apps into categories such as Action Games or Entertainment. Upon installing the application, a message alerting a user of permissions accessed appears. All apps are required to be digitally signed with a certificate before they can be installed from a marketplace. This ensures it is not possible to change the permissions in the AndroidManifest.xml file after the user has installed the app. If the APK file is modified after signing, the signatures will not match and the app will terminate.

Apps are not manually reviewed for quality or security. In response to increasing malware in the Google Play store, Google created a tool called Lockheimer which has lead to a decline in malware [256], however it has been found that it is possible to circumvent [257].

## 7.3 Android Privacy Protections

Recent research on privacy is focused on the sensitivity of data stored on mobile devices. No current platform has achieved a good balance between control, information and inter-

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_GPS" />
<uses-permission android:name="android.permission.READ_OWNER_DATA" />
<uses-permission android:name="android.permission.READ_CALL_LOG" />
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
```

Listing 1: Standard Android Permissions

```
<uses-permission android:name=
``com.google.android.providers.gsf.permission.READ_GSERVICES" />
```

Listing 2: Custom Android Permissions

activity [258]. Early protection techniques used frequent intrusive popups requesting a user for permission to access data. For example each time a location sample was required, the user is asked to give permission. Our solution only alerts users when an app is detected as an outlier. Therefore, it reduces the amount of popups from the traditional approaches.

Many users do not easily comprehend the implications of granting third party apps permission to access data [174]. A user study was performed via an Internet survey with 308 participants. It was found only 15% of the participants paid attention to the permissions at installation time. This highlights the need for improving user comprehension. PrivacyPalisade uses clear examples of the implications of permissions in outlier apps.

Self-organizing maps (SOMs) were used to visualize the Android permissions system and provide insights of where it could be improved [259]. Some permissions are used frequently while other permissions are only used by a small subset of applications. While there are many permissions a user can allow or disallow for mobile apps, it was found that few clusters cover most users' privacy preferences [260]. This can help in determining how strict to implement user privacy controls.

Risk signals have been proposed using a SVM model that compares apps across two datasets and looks for rare permissions across categories [261]. Privacygrade.org [180] researchers use crowd-sourced data which require extensive data collection. Rules for detecting dangerous combinations of permissions was proposed in [262]. For example, a combination of Internet and microphone permissions enable the app to record mobile conversations. Risk signals are also used by PrivacyPalisade to alert users of privacy implications.

A framework for detecting Android malware based on permissions was proposed in [251], where k-Means clustering is combined with decision tree learning. Malware samples were used to train the classifier. Similarly PUMA [252] compares extracted permissions from an APK and compares them to known malware samples provided by the VirusTotal online security tool. Crowdroid [253] also uses k-Means to detect malware, however instead of using permissions data, crowd-sourced samples of user behavior related to system calls is used. These approaches are aim to use requested permissions to flag potential Malware applications and not legitimate applications that are privacy invasive.

AndroidLeaks [171] and Stowaway [9] decompile the Java source code and look for methods that pass personal information and detect overprivilege in apps. It has been determined that many apps do not follow the least path of privilege. While this is useful in detecting what data is captured from an app, it does not provide information on whether the collection of data was justified to provide functionality.

TaintDroid attempts to provide insights into how apps use and share data by providing continuous real time monitoring of data and when data leaves the phone [241]. While TaintDroid provides a good monitor of what data is leaving the phone, it does not provide any protection to prevent it from occurring.

## 7.4 Problem Definition

Given an app $a$ in a set of apps $A$, there also exists a set of similar apps denoted $SA(a)$. There exists a set of permissions $P$ that apps request a subset of these permissions to access protected smartphone functionality. Each app $a \in A$ specifies a set of permissions returned by the function $Perms(a)$ where $Perms(a) \subseteq P$. The aim is to detect if an app $a \in A$ requests permissions $Perms(a) \in P$ that are not outliers relative to similar apps $SA(a) \in A$.

## 7.5 App Classification

In order to detect if an app has permissions that are excessive relative to advertised functions, a comparison can be made with apps that provide comparable features. Similar apps as suggested by marketplaces provides a good cohort for comparisons. Anomaly detection techniques like Isolation Forest can be trained on permissions of similar apps and evaluated on the target app to determine if it is an outlier.

### 7.5.1 Dataset

Apps in the Google Play marketplace are listed under a range of categories. Each detailed app listing states all permissions required, also presented are suggested apps that are similar. We developed a web scraper to collect this information across 16,581 popular apps. While Google Play contains millions of apps, scrolling through the catalog lists the most popular apps. Thus, we considered the popular apps interesting for analysis because they are widely used.

| Metric | Description |
|---|---|
| App Name | Developer Google Play name |
| App Package Name | App Java package name |
| App Permissions | Summarized permissions |
| Google Play Category | Developer selected for app |
| Similar Apps | Suggested similar apps |

Table 7.1: Metrics Collected from the Google Play Store by Our Web Scraper

| Permission | Count | Count (%) |
|---|---|---|
| Full network access | 16,089 | 97.00 |
| Read phone status and identity | 7,688 | 46.37 |
| Approximate location | 3,839 | 23.15 |
| Precise location | 3,703 | 22.33 |
| Run at startup | 3,018 | 18.20 |
| Read your contacts | 1,495 | 9.02 |
| Record audio | 1,325 | 7.99 |
| Directly call phone numbers | 867 | 5.23 |
| Send SMS messages | 481 | 2.90 |
| Read your text messages | 458 | 2.36 |
| Read calendar events | 386 | 2.32 |

Table 7.2: Apps Requiring Access to Sensitive Permissions

### Google Play Web Scraping

The most popular apps in the Google Play marketplace are listed in a range of categories. A user can also filter to see either free or paid apps. We developed a web scraper to collect this information.

The total amount of apps we acquired is 16,581. While the Google Play store contains many more apps, scrolling through the catalog lists obtained information on the most popular apps downloaded by many users daily. Table 7.1 lists the metrics we obtained.

To collect this data, we implemented a Python web scraper using BeautifulSoup[2] which is a framework commonly used to parse HTML content and extract key information from markup tags. The scraper visits each Google Play category and waits for the tile view of apps to load. It will then proceed to programmatically scroll down the page until no more additional apps are returned. By keeping track of the hyperlinks of the app tiles, the scraper will then follow the hyperlink to each app's detailed listing page.

When the scraper visits a detailed listing page and look for HTML tags that contain the app title and app category data. The app package name can be seen in the URL of the page

[2]BeautifulSoup: `https://www.crummy.com/software/BeautifulSoup/`

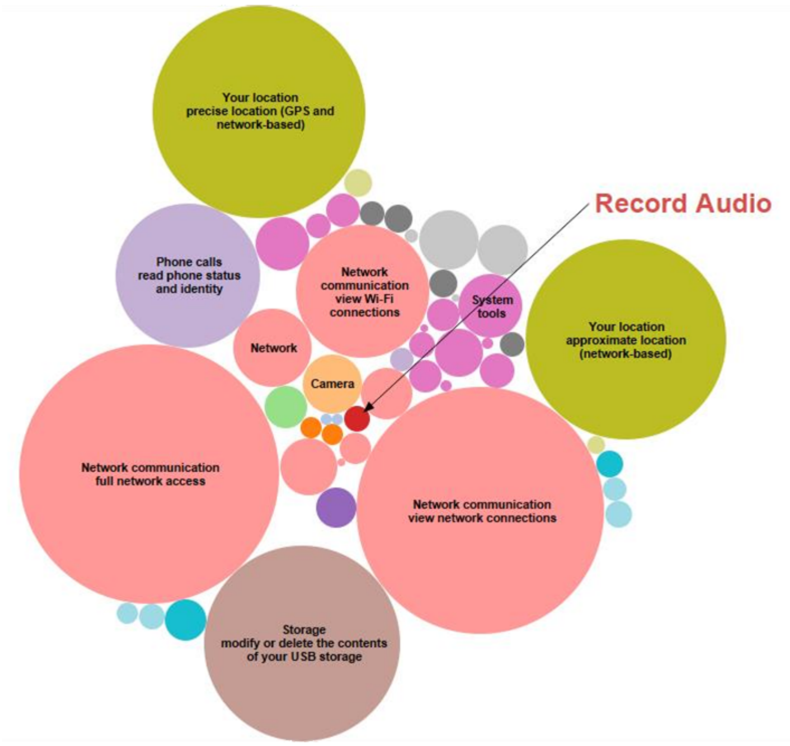



Figure 7.2: Bubble Chart of Weather App Permissions

itself. Similar apps are also captured at the bottom of the page. To obtain permissions information, there is a *"View Permissions"* hyperlink at the bottom of the page, following this hyperlink displays a popup. The tags containing the permissions information are captured.

Upon the scraper completing the parsing of the detailed listing page, all captured data is stored in a MySQL relational database.

### Google Play Scraping Results

Certain permissions give access to more sensitive data than others. Listed in Table 7.2 are permissions we consider invasive as well as the percentage of apps that are using the permission in question in our dataset.

The *"full network access"* permission was requested by almost all apps, this combined with any permission that allows the app to retrieve data, creates the possibility of an app to collect data on a network server.

Half of the apps request to *"read phone status and identity"*, which givesthe app access to the IMEI number. IMEI is a unique number that is used by many developers to track

| App Name | Perm1 | Perm2 | Perm3 | PermN |
|---|---|---|---|---|
| SimilarApp1 | 0 | 1 | 1 | 0 |
| SimilarApp2 | 1 | 1 | 1 | 0 |
| TargetApp | 0 | 1 | 0 | 1 |

Table 7.3: Representation of Apps Permissions

| $\epsilon$ | Green(%) | Blue(%) | Red(%) |
|---|---|---|---|
| 0.5 | 26.51 | 36.62 | 36.87 |
| 0.6 | 35.14 | 50.48 | 14.38 |
| **0.7** | **36.45** | **56.34** | **7.21** |
| 0.8 | 37.19 | 60.09 | 2.72 |
| 0.9 | 37.56 | 62.44 | 0.00 |

Table 7.4: Percentage of Apps Flagged Across Alert Levels for Different Values of $\epsilon$

a user. Furthermore this permission also allows the app to know when the phone rings or when a user makes a call. Almost a quarter of apps request access to location services which can be used to infer where a user lives and works. Highly sensitive permission requests to contacts, messages, calendar and microphone access were found to be minimal (less than 10%).

Filtering the data to look at specific categories gives an indication of what type of permissions are required for app functionality. Obvious permissions required by a standard weather app would include location and network access to download data from weather services. Recording audio and reading the phone status identity are not as justifiable.

While weather apps seemingly may be innocent, it is quite common for weather apps to be opened every single day by a user. The bubble chart in Figure 7.2 illustrates a bubble for each permission. The size of the bubble indicates how many apps in the Weather category use that permission. Thus, small bubbles indicate less commonly used permissions, as can be seen for *"record audio"*. Thus, if a company interested in collecting user data wants to make an app for this purpose, embedded data collection functionality inside a weather app would be a good target.

### 7.5.2 Data Mining Approach

Each app $a \in A$ requests permissions $Perms(a) \in P$. Each permission $p \in P$ is represented as a separate field using a binarized value of 1 to flag the permission is required while 0 indicates it is not required. A table is created for each target app. An example is shown in Table 7.3.

The permissions we used for training include:

- Read your contacts
- Read phone status and identity
- Approximate location
- Precise location
- Run at startup
- Record audio
- Call phone numbers
- Send SMS messages
- Read SMS messages
- Read calendar events
- Require full network access

In our method, the set of similar apps $SA(a) \in A$ is simply the apps classified in the same category as $a$. An Isolation Forest is constructed from the similar app vectors. The target app is then evaluated resulting in a IsolationScore between $0$ and $1$. A score of $1$ indicates definite anomalies, while $0.5$ indicates the app is consistent with similar apps. The following rules are applied to determine the level of severity of an app:

- Red Alert - If an IsolationScore is greater than $\epsilon$ and uses a sensitive permission;
- Blue Alert - If an IsolationScore of less than $\epsilon$ and uses any sensitive permissions;
- Green Alert - If an app does not require any sensitive permissions.

The value for $\epsilon$ used by PrivacyPalisade was empirically determined by testing different values and taking the ratio of alerts that appeared the most reasonable, in the case of our dataset it was $\epsilon = 0.7$. Table 7.4 demonstrates the effect when varying $\epsilon$.

### 7.5.3 Outlier Results

We ran our app classification technique on every app in our dataset. Table 7.5 lists common categories and the percentage of alerts triggered by apps. Communications had the highest percentage of red alerts, while books were amongst one of the safest categories.

## 7.6 PrivacyPalisade

In this section, we present the design of PrivacyPalisade, a system designed to help protect users from potentially privacy invasive apps and improve user comprehension. To improve

| Category | # Apps | Green(%) | Blue(%) | Red(%) |
|---|---|---|---|---|
| Communication | 381 | 14.70 | 73.32 | 12.07 |
| Social | 382 | 24.35 | 65.18 | 10.47 |
| Music Games | 320 | 59.06 | 30.94 | 10.00 |
| Action Games | 487 | 32.03 | 58.52 | 9.45 |
| Adventure Games | 447 | 39.60 | 54.36 | 6.04 |
| Lifestyle | 318 | 42.09 | 52.53 | 5.38 |
| Books | 356 | 55.62 | 39.89 | 4.49 |

Table 7.5: Number of Outliers Detected per Category by PrivacyPalisade

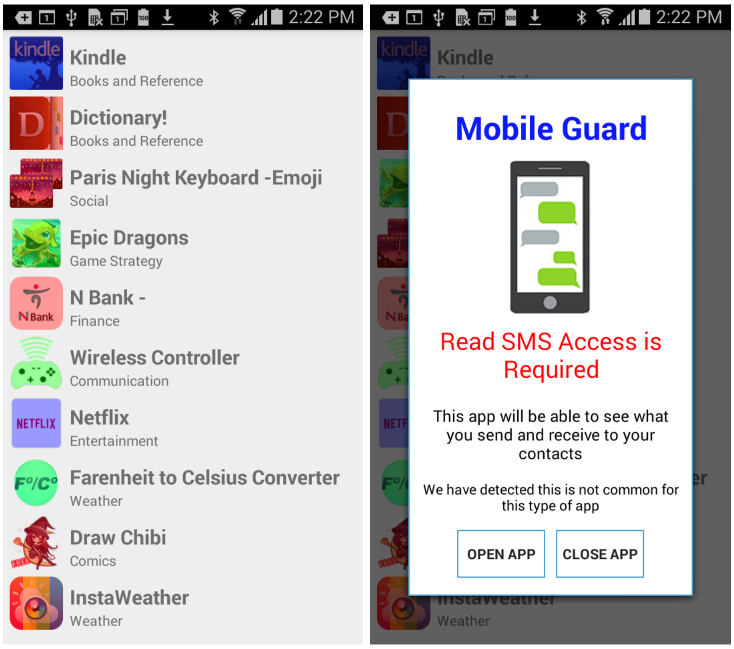
user comprehension, our system flags outlier apps installed and displays permissions implications at launch time. This is challenging to implement because third-party apps do not allow to intercept and block an app when it is launched. Thus, we made enhancements at the OS level. The Android OS Launcher is modified to overlay privacy information and recompiled as a custom ROM that can be deployed on Android devices. The system consists of a web service, an Android app and an Android background service.

The web service maintains a database of Android apps and their respective privacy information as generated in Section 7.5. By passing an Android app package name via a GET request, a JSON response is returned with the privacy rating and outlier permissions. The web service was implemented in PHP and deployed on Apache. The web service maintains a database of all Android applications and their respective privacy information as generated by our approach described in Section 7.5. Running our approach directly on Android would be computationally expensive, thus we opted for remote processing.

PrivacyPalisade runs a background service on the Android device that communicates with a server retrieving information about apps a user has installed. This information is stored in an internal database local to the user's device, entries are added and removed when a user installs and removes an app.

Our background service communicates privacy information both to the PrivacyPalisade Activity and the Android Launcher. Other apps and widgets can also use PrivacyPalisade information. Users can browse the privacy information for installed apps using the user interface displayed in Figure 7.3. Icons are colored based on invasiveness. Green denotes safe, blue indicates neutral and red for potentially invasive. When an app is opened, a popup dialog is loaded which presents a view explaining permissions used.

The Android Launcher displays the home screen, phone dialer, messaging and app icons for users to launch third-party apps. While the PrivacyPalisade UI is useful to display if




(a) List of Installed Apps (b) Visual Alerts

Figure 7.3: PrivacyPalisade UI to Alert Users of Apps Requesting Excessive Permissions

apps are safe, it is more convenient for the user to see this information directly in the launcher. To build the customized launcher, we downloaded the OS source code for Android 7.1.1 Nougat from `https://source.android.com` and compiled it on Ubuntu Linux 15.04. The custom operating system was deployed on a LG/Google Nexus 5X.

The original launcher was modified to listen for broadcasts from the background service. Based on the invasiveness level; the Bitmap of the app icon is overlaid with a color filter. An example is displayed in Figure 7.4. An additional class is added to the source tree to display custom dialog boxes when an app is executed from the Launcher application. If the user selects *"Open App"*, the original Intent to start the Activity is called, otherwise if a user selects *"Close App"*, the app will not open and the dialog will dismiss.

We show the information of permission outliers with intuitive icons. If a location permission is detected as an outlier, the user is displayed an image of a map and a house marker (Figure 7.3a) making it obvious that the app can possibly know a user's home location. The SMS messages bubble (Figure 7.5b) indicate the app wants to read SMS messages, using bubbles is intuitive inspired by the way that users typically read messages on a modern smartphone. The microphone screen displayed in Figure 7.5c makes it clear the phone can start recording.

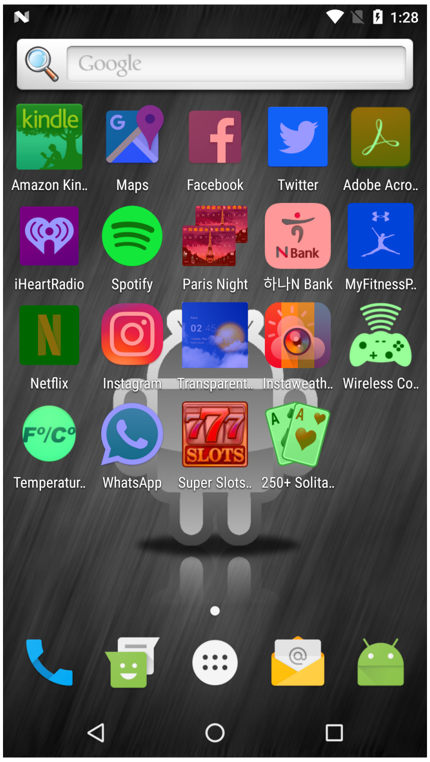



Figure 7.4: Android OS Modification to Color Code Launcher Icons

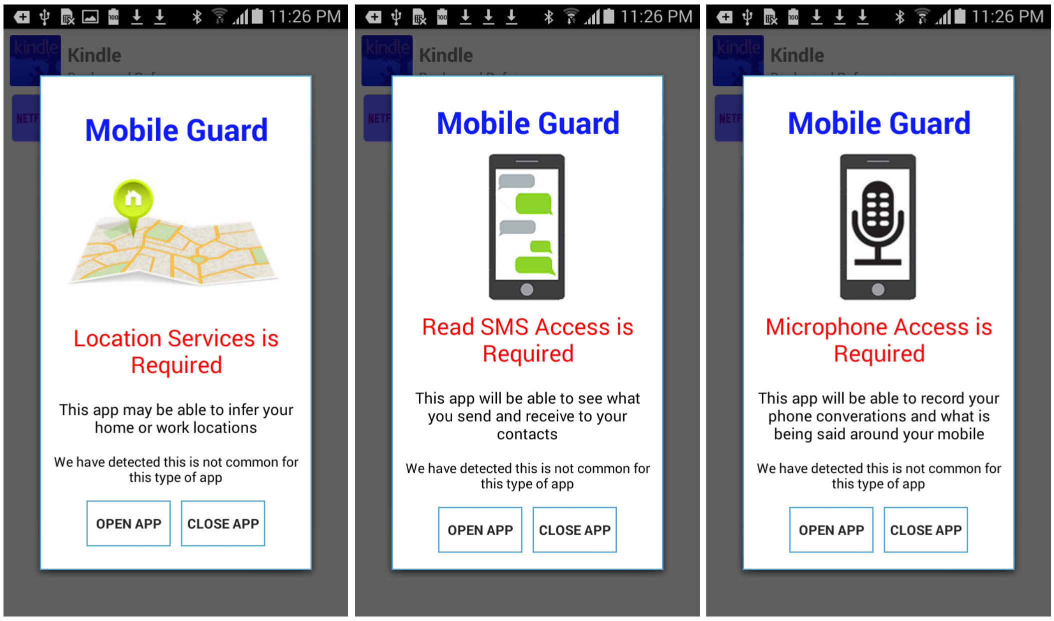
## 7.7 App Case Studies

Our study shows that PrivacyPalisade detects between 5% to 10% of apps as outliers in each Play Store category. The apps we selected to present as case studies are very popular and require permissions that are not needed for their functionalities. It was found that many apps request *"precise location"* when only *"approximate location"* is required. Furthermore, some apps request permissions in which there is no direct use case to provide app functionality. We show some examples as follows.

### 7.7.1 iHeartRadio

iHeartRadio is a popular free music streaming service for Android and iOS. At present it has received between 10 to 50 million downloads from Google Play.

PrivacyPalisade flagged it as an outlier because it required the *"precise location"* permission which only 14% of similar apps required. This permission is used to determine what local radio stations are available in the area. For iHeartRadio to perform a local ra-




(a) Location Alerts (b) Read SMS (c) Record Audio

Figure 7.5: PrivacyPalisade Sensitive Permissions Alert Dialogs

dio station lookup, *"precise location"* is not needed. TuneIn radio which provides a similar service only required *"approximate location"* to achieve the same function. All competitors received a similar number of downloads to iHeartRadio, indicating that precise location access is not a deterrent for users.

### 7.7.2 Dictionary.com

Dictionary.com provides a free online English dictionary app for its Android and iOS users. The Android version has received between 10 to 50 million downloads. The app requests 11 permissions, one of which is sensitive and flagged by PrivacyPalisade (*"precise location"*).

The app requires location to support the local lookups feature which allows the user to see nearby word searches. Similar competitive apps such as the Oxford Dictionary of English and Merriam-Webster do not require any location information. Each of these dictionary apps are very popular, thus indicating it is not apparent to users that an alternative dictionary that is not location invasive is available. Precise location is not needed to provide a nearby search feature, approximate location is sufficient and would result in the app following the least privileged path.

### 7.7.3 Hana Bank

Hana Bank is one of the largest banks in South Korea. A mobile app is provided to their customers available on both Android and iOS. Google Play states that it has received between 1 to 5 million installs. PrivacyPalisade detected 25 permissions requested, many of which give access to sensitive data. For example, only 20% of similar apps required access to the *"read your contacts"* and *"read call log"* permissions.

The app requires access to contacts to allow the user to see a list of people they can transfer money. We observed banking apps require a combination of *"directly call numbers"* and *"write call log"* for the app to provide a direct way of calling customer support numbers. Removing the *"read call log"* permission would not hinder this functionality. Many similar banking apps require the *"receive text messages"* permission which allows the app to verify the phone number via reading a confirmation SMS. For this type of verification, *"read your text messages"* is not needed. However, it is requested by the app.

## 7.8 Protection Against Sensitive Inferences

Throughout this thesis, we demonstrated how sensitive inferences can be made from seemingly innocuous data. In this section, we describe how PrivacyPalisade can prevent such inferences from being made.

PrivacyPalisade protects smartphone users by alerting them to privacy dangers of certain apps at launch time, allowing the user to make more informed decisions. A user can also decide based on the color coded information provided whether to continue using a particlar app or find a more privacy aware alternative.

The Android permissions system ensures that access permissions must be granted ahead of time before an app can make method calls via system APIs. These system method calls may be used for the purposes of accessing certain functionalities like turning on Bluetooth or retrieving sensitive data including users contacts. PrivacyPalisade running at the operating system level checks for permissions apps are requesting to determine the possible data metrics an app can access at load time.

Location services provided by Android is used in apps that require positioning estimates or location query results. We demonstrated that services that make use of continuous location queries can infer a user's traveled route in Chapter 3 and their personnel encounters in Chapter 5. Before an app that makes use of location services is executed, the dialog in Figure 7.5a is presented to the user. The visual indicates to the user what information can be inferred. Since the attacks rely on the apps being continuously used, a user can decide to use the app less frequently to ensure any data sent to a service is sparse.

Users leaving Bluetooth on all the time opens up the possibility for the location of a smartphone user to be determined indoors or have contextual information determined of when certain Bluetooth enabled devices are being used. We demonstrated how this can be performed in Chapter 4. Many apps switch on Bluetooth automatically in order to facilitate communications with other devices, users may not be aware of this. Bluetooth access is considered a sensitive permission and a user is alerted when it is accessed by an app. Since the user is aware that it has been switched on, the user can then decide if they wish for Bluetooth to remain on after they close a certain app.

As a result of learning that a user can be identified from diagnostic data as demonstrated in Chapter 6, access to any settings related permissions is considered sensitive by PrivacyPalisade and a user will be alerted. Diagnostic data is sometimes used by app developers to improve services so the user can decide if they trust the developer with this information.

## 7.9 Conclusions and Future Work

In this chapter, we evaluated the most popular Android apps and presented an approach to highlight outliers by using the permissions information of apps of similar functionality as inputs for the Isolation Forest anomaly detection technique. Privacy focused UI enhancements to Android were also demonstrated. By color coding the launcher icons, it can be easily seen by the user which apps are privacy invasive. The dialogs displayed when an outlier app is opened would help the user easily understand which apps are more privacy invasive.

We are currently in the process of deploying PrivacyPalisade on Google Nexus devices and conducting a user study. In future work we aim to implement a continuous monitoring solution to create a profile of each data entry an app has requested and sent to a server.

# Chapter 8

# Conclusions and Future Directions

In this thesis, we developed techniques to analyze smartphone data that at first glance appears innocuous, however upon further inspection can reveal sensitive information about an individual. We also presented a prototype system that makes modifications at the OS level on Android to help users better control what types of sensitive information can be accessed by third party apps. In this chapter, we provide a summary of the research contributions, the implications they have for smartphone privacy and future directions for research in this area.

In Chapter 2, we covered research highlighting the types of sensitive data stored by smartphones, associated smartphone privacy implications, generic dataset privacy protection models, privacy preserving data-mining techniques and legislative frameworks. Smartphones collect and store personal information about a user including contacts, messages, emails, social networking information, appointments, web history and location tracks. There is also data stored and sent to services that is not considered sensitive such as diagnostic data or query results. This data is accessible via third party apps and sent to cloud storage providers.

There has been little research conducted into the privacy implications of smartphone data that is more readily exchanged prompting the need for this work. Throughout this thesis, we explored data that is not considered to be sensitive and in some cases even public and demonstrated how sensitive inferences about smartphone users can still be made.

Smartphone users are becoming more aware of the potential privacy implications associated with interacting with the device. Recent studies suggest that a majority of users do value their digital privacy and are willing to pay a premium if privacy aware alternatives are available. An issue users currently face is there are not many privacy aware alternatives for smartphone platforms and third party apps and studies show users apprehensively trade their privacy for convenience.

The research conducted in this thesis is timely with public concern increasing over the privacy implications of smartphone usage. Anonymized data is often exchanged for research purposes and smartphones are a great source of certain types of individual data. Furthermore, advances in data mining has led to additional data being collected for the purpose of modeling. An understanding into the possible inferences that can be made is important to ensure both that data can be exchanged with strong safety guarantees and privacy controls to help users select the level of privacy they require can become more readily available. Insights into what seemingly innocuous data can reveal will assist smartphone designers to build in better privacy protections directly into their respective platforms.

Smartphones are maturing as a technology, with privacy as key topic it will be expected that measures are taken. Recent examples of attacks have led to adaptations being made. Our research can assist smartphone developers to further safeguard privacy by adopting the protection techniques we proposed in this thesis.

## 8.1 Summary of the Research Contributions

Query results are exchanged often amongst data providers. In Chapter 3, we demonstrated how the location of an individual and the route they traveled along could be inferred. We developed an algorithm that given a set of timestamped POIs, will return the likely route a user travels along. This was performed by first constructing a Voronoi diagram with each point representing a POI. Paths were incrementally generated to reach each Voronoi cell. A maximum speed bound is assumed based on the average speed of the road network in the area being considered. Any paths that do not reach a Voronoi cell a user resides in is discarded. A candidate path selection algorithm was then used to pick a path from the set. We proposed two candidate path selection methods. The first method is weighted to select a path based on the most central roads. The second is weighted based on the roads users travel along more frequently. It may be possible to infer other types of data from query results. For example, places that a user may frequent often.

Using just diagnostic data we demonstrated how users can be uniquely identified. We took a set of features including manufacturer, storage space, user settings and trained a naive bays classifier. Once trained, we were table to uniquely identify users in the set using a classifier on diagnostic data where the user ID was not known. Only three days of consecutive updates of diagnostic data were required to make accurate inferences. Diagnostic data is commonly shared by many apps and services and sent frequently to cloud storage services. It is possible to detect when users are using certain services with this information. Users may not want to disclose their usage behavior of apps. This data can also be linked to usage data of other apps which will in turn then be able to be used to identify them. How diagnostic data is sent needs to be reviewed. We suggest that dynamic features such as memory size are not sent to reduce the probability of being able to uniquely identify a user.

In simply scanning for Bluetooth signals, we were able to locate users indoors. Bluetooth has long been a concern for privacy researchers and many attacks to privacy have been successful like retrieving a user's address book. We developed a localization scheme that combined the benefits of both range-based and range-free methods to locate Bluetooth users with accuracies of up to 1m. A user may be comfortable with disclosing their workplace but not the exact part of the building they are located in. With our scheme, it is possible to find these users. How Bluetooth signals are transmitted and how often the device is in discoverable mode needs to be carefully considered. Even if a device is not in discoverable mode, it can be detected if a device is in range if the MAC address is known. Limiting the distance Bluetooth signals can travel is one possible solution. Therefore, only devices that are within close range can communicate with the device.

Many apps and services are making use of continuous queries. We were able to infer user encounter patterns given access to continuous location data. We were also able to make these inferences in real-time for millions of people in a city area. Proximity information between users is required to mine for these patterns. Calculating the distance between each combination pair would yield too many computations. Nearest neighbor algorithms can also be applied to this problem, however it is not efficient as only the people in proximity are required and not necessarily the nearest neighbor that is far away and not in proximity. We proposed the use of a $c$-NN query that only considers neighbors that are within a fixed threshold. By building a spatial index, taking advantage of properties that are unique to encounters, we discovered an efficient way to mine for these patterns. In discovering encounter patterns, users that are concerned that their activities are being learned should mask their location to make it harder to track.

We also proposed a protection system called PrivacyPalisade. The system attempts to balance the functionality requirements of apps with the privacy requirements needed. By using Isolation Forests, we were able to detect apps that requested permissions that were outliers in the cohort. The system is integrated directly into the operating system and retrieves privacy scores of apps installed on the smartphone. Users are warned if an app is detected as suspicious before the app is executed. In addition, the launcher icons were color coded. Using this system, we found that many apps in Android marketplaces are not privacy aware. PrivacyPalisade enables users to make more privacy conscience choices for apps.

## 8.2 Implications of Research and Suggested Safeguards

Throughout this thesis, we have highlighted how sensitive data can be inferred from data considered to be non-sensitive. Data aggregation organizations and third party apps that collect this data may assume the data is anonymous. We show that upon careful inspection that this is not the case in many instances. Mobile data can be combined with external data sources and aggregated to build a very accurate profile about an individual. In order to develop effective techniques to protect user privacy, a clear understanding is needed into what data can be sourced by an adversary as well as what data can be dangerous to a user if leaked.

Our work suggests that extra care needs to be taken by data providers to ensure data is anonymous and additional information cannot be inferred. In addition, smartphones need to be equipped with more tools to help safeguard user privacy. As the public continue to learn about privacy implications, there will be demand for more secure phones.

Legislative frameworks exist to protect user data ensuring that it is stored anonymously but they do not extend to implicitly inferring sensitive information. We believe that by the work in this thesis demonstrating cases where implying sensitive information is possible, additional regulatory guidelines can be proposed by legislators to ensure additional measures are taken to protect user data.

Spatial nearest neighbor query results often yield locations that are close to the position of the initiator. We demonstrated how a smartphone user's route can be reconstructed to a high accuracy using only the query results without access to the original positioning estimates. Query results retained by location services should be a fair distance away from where the query was initiated to help ensure their user's privacy.

Devices that make use of radio technologies such as Bluetooth and WI-FI make available signal strength information without any authentication having to be performed. We demonstrated that it is possible to infer indoor location information and contextual information about a user. Techniques to suppress, generalize or add noise to the signal information should be considered. For most applications such as signal strength indication bars, only granular signal strength information is required. These protections would make it much more difficult to infer user location.

When smartphone users reveal their location, many are unaware of the possibility of deriving further contextual information. We demonstrated that the encounters of individuals can be detected. As a result, smartphone platforms should take care to not send location information to providers when the user is within proximity to people in their social network. This protection would make it far more difficult to detect encounter patterns.

Certain metrics considered to be non-sensitive such as diagnostic data are personally identifying. In just examining the raw data, it may be hard to find user insights, with data aggregation and data mining, more information can be revealed. For example, we revealed how diagnostic data can be used to identify an individual and thus, should be considered sensitive. In turn, leading diagnostic data to being considered sensitive in own work when developing PrivacyPalisade.

This research helps provide a far better understanding of what data can be captured, stored and insights inferred of person or a group by only using their smartphone data. In also showing that modifications to the smartphone platforms operating system via PrivacyPalisade can further protect privacy, smartphone developers can adopt these methods to better safeguard user privacy.

## 8.3 Future Directions

Smartphone privacy research has recently received a lot of attention. In this section, we highlight a few possible future directions.

In this work we considered a few seemingly innocent data features and went through great efforts to infer sensitive information. While careful inspection is required, we showed it is possible to infer this sensitive information such as traveled routes and the encounters individuals have with people. For future work, additional metrics that are considered innocent can be inspected and analyzed to determine if additional information can be derived. One example of a possible direction is to determine if a combination of sensors such as temperature and light can be used to derive context of the environment the user is likely within e.g. indoors or outdoors. This knowledge will continue to help in refining privacy protection mechanisms.

In terms of protection, our system stops users from using apps that are considered suspicious. Integrating directly into the operating system opens up the doors to more sophisticated protection techniques. A future direction for this is to analyze traffic being sent to services and detecting the probability of attacks occurring. If the system detects data sent can lead to a sensitive inference being made, then it can be blocked. Furthermore, a profile of what data is sent to what app can be built and displayed to the user so they have an understanding into what third parties know about them. In many cases, finding a privacy aware app may not be an option. Another possible direction is to investigate how to best feed apps obfuscated data so the app can still continue to function while not compromising user privacy.

Third party app developers can implement apps to access data. An investigation is needed to determine developer comprehension of permissions used to request data. There have been studies performed investigating user comprehension of smartphone permissions but less focus on the developers. There are often less data invasive paths to perform required functionality. For example, when performing a local radio search only approximate location data is needed, not precise. Promoting developers to better carefully consider privacy applications can lead to more secure apps and thus, providing users with more privacy aware alternatives to common apps they use.

User comprehension of permissions has been demonstrated to be low (see Section 2.7.4). Further techniques are needed to assist users construct privacy protection profiles. Depending on the type of user and their privacy requirements, profiles need to be constructed. Some users may have stricter privacy requirements and are willing to sacrifice more privacy than others. Further studies are needed to show how to best customize user profiles and provide a guarantee to users that their privacy requirements set in the profile are being met by the platform.

While it is widely known that it is possible to easily collect smartphone information via third party apps and that data mining is being performed, few studies consider in detail the motivations. More work is needed to better understand the types of organizations that are mining for user smartphone data, the methods that are commonly employed to acquire said data and for what purposes. Enjoyed benefits from data mining vary depending on the sector data analysts are working within. In better knowing the purposes, it will be easier to establish privacy models that can achieve the goals of the organizations without compromising user privacy.

Users install multiple apps on their smartphones and make use of many different services. A single app or service may not necessarily maintain all of a user's mobile data. For example, a navigation app may maintain location information about a user while a media streaming app may maintain music preferences. It is possible that data aggregation techniques can be employed making use of multiple data sources to maintain a user profile. An investigation is needed into the extent of how effective data aggregation across multiple data sources is and how to protect against it. An even greater understanding can also be gained from combining internal smartphone data with external sources such as a national census. More work is needed to understand how data features can be combined and what could be learned as a result.

Following on from our results, practical applications could follow including security software can be better implemented to protect user privacy and better manage the exchange of smartphone data. We hope to raise user awareness of the potential dangers of certain services and promote stricter privacy models for the next generation of smartphones.

# Appendices

# Appendix A

# Capturing Smartphone Data

Mobile smartphones store a significant amount of information about a user. This data can be easily accessed using standard API calls commonly provided by popular platforms. In this chapter, we present the technical details of how to capture data and send it to a remote server. Our primary focus was on the Android platform which has the highest number of users worldwide and is accessible for research purposes since it is open source.

The aim was to explore if stored mobile data can be easily captured via third party apps commonly found in app repositories. When we prototyped our application, literature on the issue was scarce. There are now numerous articles discussing this issue in detail and there have also been media publications bringing attention to the general public. Modern platforms aimed to provide a rich environment for developers to create apps on the smartphone, however in doing so have opened up a new attack vector for those who wish to collect user data.

*MobileSensor* captures data stored natively on the device and sends it to a remote database. Publicly available APIs that retrieve data and pass it to the app were used. It is important to note that no exploits were utilized in the data collection as the focus was on what standard apps are capable of accessing. Data captured is stored in an internal relational database and periodically synchronized with a remote database server. Modules implemented in this work have been reused in varied research completed in the department where data collection via smartphones is required. This appendix chapter can serve as a guide to providing best practices when making use of smartphone data.

## A.1 Introduction

Android is a popular open source mobile platform. We focused on Android smartphones because its open nature lends itself to being suitable for research purposes as it can be customized to suit a wide range of needs. Android, Inc was founded in 2003 with the aim of developing smarter mobile devices. The Android OS is not limited to mobile devices and has been used on devices including tablets, e-readers, car computers and media centers.

The initial goal of the then startup was to compete with Nokia's Symbian based operating system and Windows Mobile. Google acquired Android, Inc in 2006 and invested heavily in Android. When Android launched, its main competitor in terms of functionality was the iPhone. Symbian lagged behind in software features offered. While Android and iOS provide similar functionality, the software architecturally is vastly different. This lead to mobile app development companies having to create two different versions of the same product.

The first commercial phone was made available by HTC in 2008. Android applications do not run in the Java Virtual Machine but rather a specialist virtual machine called Dalvik. Standard Java code can be used in Android apps and are built on top of the Android API [1]. XML is heavily used in Android development for application properties, user interface designs and values for variables. Every Android application has an *AndroidManifest.xml* file that stores common meta-data including Android API version required, user permissions that are required to be granted and features the app uses.

Android applications can be developed on Windows, Mac OS X and Linux via Android Studio. Each major release of Android has an associated API Level. Higher API levels support all the functionality of lower API levels as a general rule. For example, code developed for API level 14 would work fine on API level 18. Many devices have much older versions of Android running on them so developers typically target lower API levels to gain good device coverage for a deployed app. Android devices support upgrading the operating system to the latest version the manufacture supports so it is difficult for users to install the latest version as proprietary drivers are often required.

[1] Android API: `http://developer.android.com/reference/packages.html`

### A.1.1 Activities

Android initiates code in an *Activity* instance that provides callback methods that correspond to a specific stage in the life cycle of an app. Each *Activity* needs to be declared in the Android Manifest in addition to a default launch activity. Activities are linked to a user interface displayed on the screen and are not guaranteed to run in the background.

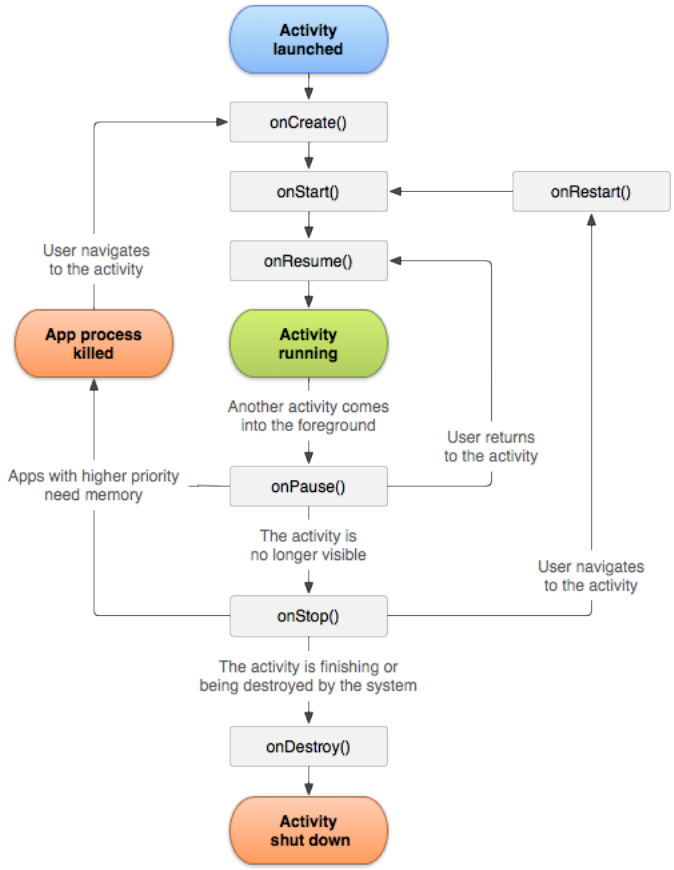



Figure A.1: Activity Lifecycle as Illustrated in the Android API[2]

### A.1.2 Services

Initialized code intended to run long running operations. Needs to be registered as a service in the Android Manifest. Can be used to run operations in the background. Communication between the Activities and Services is facilitated mainly using BroadcastReceiver classes.

[2]Android Activity API: `https://developer.android.com/reference/android/app/Activity.html`

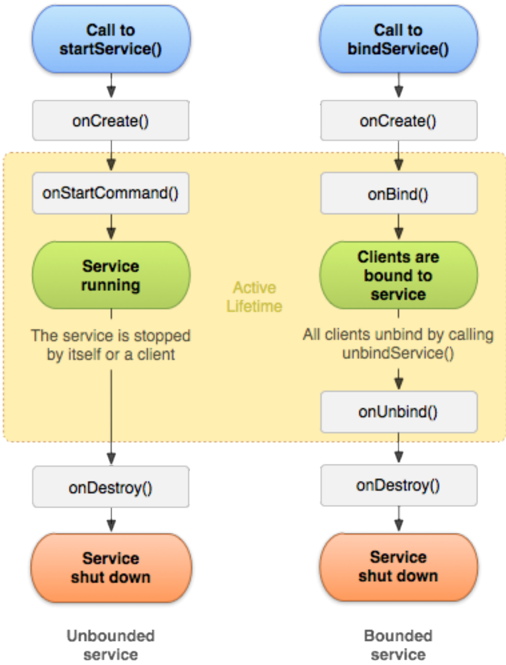



Figure A.2: Service Lifecycle as Illustrated in the Android API[3]

### A.1.3 Mobile Data Transfer

Every mobile facilitates data transfer wirelessly. Data captured from mobile sensors can be easily sent to another device for processing via the following communications mediums:

- 2G/3G/LTE Network connection
- WI-FI Network Connection
- Bluetooth Data Transfer
- SMS Messages
- Near Field Communication (NFC)
- Infrared

### A.1.4 Permissions

The Android API provides many method calls for developers to build functionality upon. Some of these method calls are sensitive and require permissions be declared in the AndroidManifest. Permissions required by an Android app can be added as follows:

```
<uses-permission android:name="PERMISSION_NAME" />
```

[3]Android Service API: https://developer.android.com/guide/components/services.html

## A.2 Data Captured

Mobile hardware is increasingly becoming more sophisticated. Mobile devices contain many instruments that can be used as sensors. The following metrics were successfully captured using standard API calls:

- GPS and tagged places
- Cell Tower and signal strength
- WI-FI devices in proximity
- Bluetooth devices in proximity
- Apps Installed and apps permissions
- App usage
- Accelerometer
- Gyroscope
- Magnetic compass
- True compass
- Orientation
- Network traffic stats
- Call logs meta data
- Contacts data
- Hardware info
- Mobile features (Camera, WI-FI, Bluetooth etc.)
- File names on SD Card
- Calendar entries
- Device language and languages of active keyboards
- Device setting preference accessible (109 variables)
- Last alarm clock set
- SMS messages
- CPU/RAM usage

## A.3 Social Media Network Data

Apps are provided by social media networks allowing for users to interact with the service conveniently on their smartphone. In many cases a user grants third party apps access to their social network. This is commonly seen at login screens where the user can conveniently login using their social network profile instead of creating an additional profile for the service provided by the third party app. This scenario is one example of how a third party app can gain access to private data not stored on the device itself.

## A.4 Sensor Data

Android devices are equipped with a variety of internal sensors that can be used to detect how the user is interacting with the device. Many of these sensors are abstracted by the *Sensor* class and values are accessible via the *SensorManager* class.

The *Sensor* class provides constants for different types of sensors that can be passed into the *getDefaultSensor* method call. Registering a *SensorListener* for the sensor will return sensor events to the *onAccuracyChanged* and *onSensorChanged* methods. The *SensorEvent* class passed to *onSensorChanged* provides the sensor values that can be then parsed to suit the app's requirements (e.g. moving a character in a game).

```
mSensorManager =
    (SensorManager) getApplicationContext().
        getSystemService(getApplicationContext().SENSOR_SERVICE);
mAccelerometer =
    mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
mSensorManager.registerListener(this,
                        mAccelerometer,
                        SensorManager.SENSOR_DELAY_UI);
public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // Accuracy is given in onSensorChaged
}
public void onSensorChanged(SensorEvent event) {
    int accuracy = event.accuracy;
    long timestamp = event.timestamp;
    float values[] = event.values;
}
```

Listing 3: Retrieving Sensor Data in Android

The following sensors are available on Android:

- Accelerometer
- Ambient Temperature
- Gravity
- Gyroscope
- Heart Beat
- Heart Rate
- Light
- Linear Acceleration
- Magnetic Field
- Motion
- Orientation
- Pressure
- Rotation
- Step Counter
- Step Detector

## A.5 Location Data

Android devices all come equipped with a GPS and can provide location samples as accurate as 5m. Location services can be accessed via the *LocationManager* class. The *requestLocationUpdates* method call can be used to register a *LocationListener* class that returns location updates via the callback method *onLocationChanged*.

```
LocationManager lm =
    (LocationManager) getApplicationContext().
        getSystemService(Context.LOCATION_SERVICE);
lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, freq, 0, this);
...
@Override
public void onLocationChanged(Location location) {
    double lat = location.getLatitude();
    double lng = location.getLongitude();
    double alt = location.getAltitude();
}
```

Listing 4: Retrieving Location Samples in Android

## A.6 Camera

Android phones usually come with a front and back facing camera Photos can be taken and then processed by Computer Vision Algorithms. Launching the camera involves starting the camera activity via an *Intent* provided by the *MediaStore* class. Once a user takes a photo, it will be returned as an Intent containing the image information passed via the *onActivityResult* call back.

```
static final int REQUEST_IMAGE_CAPTURE = 1;

Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
if(takePictureIntent.resolveActivity(getPackageManager()) != null) {
    startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
}

@Override
protected void onActivityResult(int requestCode, int resultCode,
    Intent data) {
    if(requestCode == REQUEST_IMAGE_CAPTURE &&
        resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
    }
}
```

Listing 5: Taking a Photo with the Camera

## A.7 Contacts

Contacts can be retrieved using a content resolver query with the *URI* provided by the *ContactsContract* class. Iterating over the cursor allows for each contact to be retrieved.

```
try {
    Cursor c =
        getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,
        null, null, null, null);

    while(c.moveToNext()) {
        String contactId = c.getString(
            c.getColumnIndex(ContactsContract.Contacts._ID));
        String name =
        c.getString(c.getColumnIndex(PhoneLookup.DISPLAY_NAME));
        Cursor phones = getContentResolver().query(Phone.CONTENT_URI,
                        null, Phone.CONTACT_ID + " = " + contactId,
                        null, null);

        while(phones.moveToNext()) {
            String number =
                phones.getString(phones.getColumnIndex(Phone.NUMBER));
        }
    }
    c.close();
} catch(Exception e) {
    Log.d("Exception", e.toString());
}
```

Listing 6: Retrieving Contacts in Android

## A.8 Call Logs

Call logs can be retrieved by issuing a query to the content resolver using a *URI*. Iterating through the cursor allows for call log information to be retrieved for calls made.

```
try {
     Uri cLog = Uri.parse("content://call_log/calls");
     Cursor c = getContentResolver().query(cLog, null, null, null, null);

     while(c.moveToNext()) {
        String number =
        c.getString(c.getColumnIndex(CallLog.Calls.NUMBER));
        String duration =
        c.getString(c.getColumnIndex(CallLog.Calls.DURATION));
        String timestamp =
        c.getString(c.getColumnIndex(CallLog.Calls.DATE));
     }
} catch(Exception e) {
     Log.d("Exception", e.toString());
}
```

Listing 7: Retrieving Call Logs in Android

## A.9 Messages

Text messages sent and received can be retrieved via a content resolver query using the *URI "content://sms/inbox"*. Itearting through the *Cursor* allows for retrieval of each SMS message.

```
Uri smsURI = Uri.parse("content://sms/inbox");
Cursor smsCursor = getContentResolver().query(smsURI, null, null, null,
    null);

while(smsCursor.moveToNext()) {
    String person = smsCursor.getString(4);
    String date = smsCursor.getString(5);
    String read = smsCursor.getString(8);
    String type = smsCursor.getString(10);
    String subject = smsCursor.getString(12);
    String body = smsCursor.getString(13);
}
```

Listing 8: Retrieving Text Messages Android

## A.10 Language

Languages that are used on the device can be detected by retrieving the available input methods. The *InputMethodManager* class provides the method *getEnabledInputMethodList* that returns a list of *InputMethodInfo* objects that provide information about the input type. The default language can also be determined using the *Locale* class.

```
InputMethodManager imeManager =
    (InputMethodManager) getApplicationContext().
        getSystemService(INPUT_METHOD_SERVICE);
List<InputMethodInfo> lst = imeManager.getEnabledInputMethodList();
String defaultLanguage = Locale.getDefault().getLanguage();

for(InputMethodInfo info : lst) {
    String inputLabel =
        new String(info.loadLabel(getPackageManager()).toString());
    List<InputMethodSubtype> list =
       imeManager.getEnabledInputMethodSubtypeList(info, true);
    for(int i=0; i<list.size(); i++) {
        InputMethodSubtype currInputMethodSubtype = list.get(i);
        String keyboardLocale = currInputMethodSubtype.getLocale();
        boolean keyboardAuxiliary =
            currInputMethodSubtype.isAuxiliary();
        String keyboardInputMode = currInputMethodSubtype.getMode();
    }
}
```

Listing 9: Retrieving Languages set in Android

## A.11 Apps Installed

Apps installed on the device can give an indication of user activity Android allows this information to be easily accessible via the *PackageManger* class. The method call *getInstalledPackages* returns a list of *PackageInfo* objects.

```
PackageManager pm = getPackageManager();
List<PackageInfo> packages =
    pm.getInstalledPackages(
        PackageManager.GET_META_DATA      |
        PackageManager.GET_PERMISSIONS    |
        PackageManager.GET_ACTIVITIES     |
        PackageManager.GET_CONFIGURATIONS);

for(PackageInfo packageInfo : packages) {
    String packageName = packageInfo.packageName;
    int versionCode = packageInfo.versionCode;
    ApplicationInfo applicationInfo = packageInfo.applicationInfo;
    PermissionInfo[] permissions = packageInfo.permissions;
    String applicationName = applicationInfo.loadLabel(pm).toString();
}
```

Listing 10: Retrieving Apps Installed in Android

## A.12 App Usage

Android is built on top of the Linux kernel and uses virtually the same process scheduler. Running processes can be retrieved using the *ActivityManager* class. The method call *getRunningAppProcesses* returns a list of *RunningAppProcessInfo* objects.

```
ActivityManager actvityManager =
    (ActivityManager) getSystemService(Context.ACTIVITY_SERVICE);
List<RunningAppProcessInfo> processes =
    actvityManager.getRunningAppProcesses();

for(int i=0; i<processes.size(); i++) {
    RunningAppProcessInfo currentRunningTaskInfo =
        (RunningAppProcessInfo) processes.get(i);
    int pid = currentRunningTaskInfo.uid;
    int[] pidArray = {pid};
    String processName = currentRunningTaskInfo.processName;
    android.os.Debug.MemoryInfo memoryInfo =
        actvityManager.getProcessMemoryInfo(pidArray)[0];
    int totalPrivateDirty = memoryInfo.getTotalPrivateDirty();
    int totalPss = memoryInfo.getTotalPss();
    int totalSharedDirty = memoryInfo.getTotalSharedDirty();
    int importanceFlag = currentRunningTaskInfo.importance;
}
```

Listing 11: Retrieving App Usage in Android

## A.13 Bluetooth

All modern mobiles devices have Bluetooth Bluetooth is a well-established technology supported by many different devices.

The *BluetoothAdapter* class allows for detecting if Bluetooth is enabled, if not it can be started by starting an activity provided by an *Intent* in the *bluetoothAdapter*. Calling the method *startDiscovery* returns any detected Bluetooth devices asynchrnosuly via a *BroadcastReceiver* class. The *onReceive* method is sent an *Intent* object that contains a *BluetoothDevice* object.

```
BluetoothAdapter mBluetoothAdapter =
   BluetoothAdapter.getDefaultAdapter();

if(!mBluetoothAdapter.isEnabled()) {
    Intent btIntent =
        new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    btIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(btIntent);
}

mBluetoothAdapter.startDiscovery();

private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if(BluetoothDevice.ACTION_FOUND.equals(action)) {
            BluetoothDevice device =
                intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            int RSSI =
                intent.getShortExtra(BluetoothDevice.EXTRA_RSSI,
                    Short.MIN_VALUE);
            String deviceName = device.getName();
            String deviceAddress = device.getAddress();
            int bondStateFlag = device.getBondState();
            String bondState = "";

            if(bondStateFlag == BluetoothDevice.BOND_BONDED) {
                bondState = "BOND_BONDED";
            }
            if(bondStateFlag == BluetoothDevice.BOND_BONDING) {
                bondState = "BOND_BONDING";
            }
            if(bondStateFlag == BluetoothDevice.BOND_NONE) {
                bondState = "BOND_NONE";
            }
        }
    };
}
```

Listing 12: Retrieving Bluetooth

## A.14 CPU/RAM Usage

The amount of memory android activities occupy can be retrieved via the *ActivityManager* class while the file */proc/stat* can be read to determine the CPU usage.

The *ActivityManager* class provide to retrieve memory information and store it in the *MemoryInfo* class. While direct calls to retrieve CPU usage is not available, the file */proc/stat* commonly found on Unix based operating systems provides this information which can be accessed and parsed.

```
  //Read Memory usage
  MemoryInfo mi = new MemoryInfo();
  ActivityManager activityManager =
       (ActivityManager) getSystemService(ACTIVITY_SERVICE);
  activityManager.getMemoryInfo(mi);
  long ramUsageMB = mi.availMem / 1048576L;
  //Read CPU Usage
  StringBuffer sb = new StringBuffer();
  try {
      BufferedReader procStat =
          new BufferedReader(new FileReader("/proc/stat"));
} catch(Exception e) {
     Log.d("Exception", e.toString());
}
String cpuInfo = sb.toString();
```

Listing 13: Retrieving CPU/RAM Usage in Android

## A.15 WI-FI

WI-FI devices within range that are discoverable can be easily accessed The *WifiManager* class provides a method call *setWifiEnabled* that allows for toggling if the WI-FI is on or off. The *getScanResults* method call returns a list of *ScanResult* objects that provide information about sensed WI-FI access points.

```
wifi = (WifiManager) getApplicationContext().
    getSystemService(Context.WIFI_SERVICE);
if(wifi.isWifiEnabled() == false) {
    wifi.setWifiEnabled(true);
}
results = wifi.getScanResults();
for(int i=0; i<results.size(); i++) {
    ScanResult result = results.get(i);
    //result.SSID;
}
```

Listing 14: Retrieving WI-FI in Android

## A.16 Calendar

The calendar reveals personal information about a users daily activities. This can be retrieved through the use of a content resolver query. The URI to access calender content is *"content://com.android.calendar/calendars"*. The resolver returns a *HashSet* containing all found Calendar IDs. The associated calendar entry information can then be accessed via the *Cursor* objects returned from the content resolver query.

```
Uri calendarURI = Uri.parse("content://com.android.calendar/calendars");
Cursor calendarCursor =
    getContentResolver().query(calendarURI, null, null, null, null);
HashSet<String> calendarIds = new HashSet<String>();
while(calendarCursor.moveToNext()) {
    String _id = calendarCursor.getString(0);
    String displayName = calendarCursor.getString(1);
    Boolean selected = !calendarCursor.getString(2).equals("0");
    calendarIds.add(_id);
}
for (String id : calendarIds) {
    Uri.Builder builder =
        Uri.parse("content://com.android.calendar/instances/when").
        buildUpon();
    ContentUris.appendId(builder, Long.MIN_VALUE);
    ContentUris.appendId(builder, Long.MAX_VALUE);
    Cursor eventCursor =
        getContentResolver().query(builder.build(), null, null, null,
        null);

    while(eventCursor.moveToNext()) {
        String eventEndTimezone = eventCursor.getString(0);
        String hasAlarm = eventCursor.getString(3);
        String calendarTimezone = eventCursor.getString(8);
        String startDay = eventCursor.getString(9);
        String description = eventCursor.getString(10);
        String hasExtendedProperties = eventCursor.getString(15);
        String allDay = eventCursor.getString(18);
        String organizer = eventCursor.getString(19);
        String eventTimezone = eventCursor.getString(25);
        String createTime = eventCursor.getString(26);
        String eventId = eventCursor.getString(32);
        String title = eventCursor.getString(38);
        String availability = eventCursor.getString(44);
        String maxReminders = eventCursor.getString(49);
        String accessLevel = eventCursor.getString(50);
        String calendarColor = eventCursor.getString(53);
        String duration = eventCursor.getString(54);
        String guestsCanInviteOthers = eventCursor.getString(57);
        String eventColor = eventCursor.getString(58);
        String eventStatus = eventCursor.getString(60);
    }
}
```

Listing 15: Retrieving Calendar in Android

## A.17 Settings

User settings can be retrieved via the *Settings* class. Many of the settings are toggles represented as *ints* although settings can also be represented as other data types such as *String*. All settings that are available to be retrieved are represented as constants in the *Settings* class.

```
Settings sets = Android.provider.Settings;
ContentResolver cr = getContentResolver();
adbEnabled = setts.System.getInt(cr, System.ADB_ENABLED);
airplaneModeOn = sets.System.getInt(cr, System.AIRPLANE_MODE_ON);
alarmAlert = setts.System.getInt(cr, System.ALARM_ALERT);
androidId = sets.System.getInt(cr, System.ANDROID_ID);
autoTime = setts.System.getInt(cr, System.AUTO_TIME);
autoTimeZone = setts.System.getInt(cr, System.AUTO_TIME_ZONE);
bluetoothDiscoverability =
    setts.System.getInt(cr, System.BLUETOOTH_DISCOVERABILITY);
bluetoothOn = setts.System.getInt(cr, System.BLUETOOTH_ON);
dataRoaming = setts.System.getInt(cr, System.DATA_ROAMING);
debugApp = setts.System.getInt(cr, System.DEBUG_APP);
dimScreen = setts.System.getInt(cr, System.DIM_SCREEN);
fontScale = setts.System.getInt(cr, System.FONT_SCALE);
httpProxy = setts.System.getInt(cr, System.HTTP_PROXY);
installNonMarketApps =
    setts.System.getInt(cr, System.INSTALL_NON_MARKET_APPS);
modeRinger = setts.System.getInt(cr, System.MODE_RINGER);
lockPatternEnabled =
    setts.System.getInt(cr, System.LOCK_PATTERN_ENABLED);
networkPreference = setts.System.getInt(cr, System.NETWORK_PREFERENCE);
notificationSound = setts.System.getInt(cr, System.NOTIFICATION_SOUND);
radioBluetooth = setts.System.getInt(cr, System.RADIO_BLUETOOTH);
radioCell = setts.System.getInt(cr, System.RADIO_CELL);
radioNfc = setts.System.getInt(cr, System.RADIO_NFC);
radioWifi = setts.System.getInt(cr, System.RADIO_WIFI);
ringtone = setts.System.getInt(cr, System.RINGTONE);
screenBrightness = setts.System.getInt(cr, System.SCREEN_BRIGHTNESS);
screenOffTimeout = setts.System.getInt(cr, System.SCREEN_OFF_TIMEOUT);
soundEffectsEnabled =
    setts.System.getInt(cr, System.SOUND_EFFECTS_ENABLED);
textAutoReplace = setts.System.getInt(cr, System.TEXT_AUTO_REPLACE);
textShowPassword = setts.System.getInt(cr, System.TEXT_SHOW_PASSWORD);
userRotation = setts.System.getInt(cr, System.USER_ROTATION);
vibrateOn = setts.System.getInt(cr, System.VIBRATE_ON);
volumeAlarm = setts.System.getInt(cr, System.VOLUME_ALARM);
volumeRing = setts.System.getInt(cr, System.VOLUME_RING);
volumeSystem = setts.System.getInt(cr, System.VOLUME_SYSTEM);
volumeVoice = setts.System.getInt(cr, System.VOLUME_VOICE);
wallpaperActivity = setts.System.getInt(cr, System.WALLPAPER_ACTIVITY);
waitForDebugger = setts.System.getInt(cr, System.WAIT_FOR_DEBUGGER);
wifiOn = setts.System.getInt(cr, System.WIFI_ON);
```

Listing 16: Retrieving Settings in Android

## A.18 Cell Tower

Cell tower information can be found using the *TelephonyManager* class. The *PhoneStateListener* class calls methods *onCellInfoChanged*, *onCellLocationChanged* and *onSignalStrengthsChanged* when the smartphone switches cell towers.

The network operator information can then be used to reverse lookup a database of known cell tower locations to determine an approximate location of where a smartphone user is located.

```
telephonyManager =
    (TelephonyManager) getApplicationContext().
        getSystemService(Context.TELEPHONY_SERVICE);
telephonyManager.listen(phoneStateListener,
                PhoneStateListener.LISTEN_CELL_LOCATION);
telephonyManager.listen(phoneStateListener,
                PhoneStateListener.LISTEN_SIGNAL_STRENGTHS);
PhoneStateListener phoneStateListener = new PhoneStateListener() {

@Override
public void onCellInfoChanged (List<CellInfo> cellInfo) {

}

public void onCellLocationChanged (CellLocation location) {
    StringBuffer str = new StringBuffer();
    String networkOperator = telephonyManager.getNetworkOperator();
    String mcc = networkOperator.substring(0, 3);
    String mnc = networkOperator.substring(3);

    if(location instanceof GsmCellLocation) {
        GsmCellLocation loc =
        (GsmCellLocation) telephonyManager.getCellLocation();
    }
    else if(location instanceof CdmaCellLocation) {
        CdmaCellLocation loc =
        (CdmaCellLocation) telephonyManager.getCellLocation();
    }
}

public void onSignalStrengthsChanged(
            SignalStrength signalStrength) {
    String cdmaDbm = String.valueOf(signalStrength.getCdmaDbm());
    String cdmaEcio = String.valueOf(signalStrength.getCdmaEcio());
    String evdoDbm = String.valueOf(signalStrength.getEvdoDbm());
    String evdoEcio = String.valueOf(signalStrength.getEvdoEcio());
    String evdoSnr = String.valueOf(signalStrength.getEvdoSnr());
    String gsmBitErrRate =
        String.valueOf(signalStrength.getGsmBitErrorRate());
    String gsmSigStrength =
        String.valueOf(signalStrength.getGsmSignalStrength());
}
```

Listing 17: Retrieving Cell Tower Information in Android

## A.19 Filenames

All files stored in external storage such as an SD card can be accessed. The external storage directory can be found using the *Environment* class. The list of all files in the directory can be accessed using the *listFiles* method in the *File* class. If a file is found to be a directory, then a recursive call can be made to determine files in the sub-directory. In general, these files can then be accessed.

```
filesInDrive = new ArrayList<String>();
String filePath =
         Environment.getExternalStorageDirectory().getAbsolutePath();
File file = new File(filePath);
getFiles(file);

 public void getFiles(File dir) {
     for(int i=0; i<dir.listFiles().length; i++) {
         File currentFile = dir.listFiles()[i];
         if(currentFile.isDirectory()) {
             getFiles(currentFile);
         }
         else {
             filesInDrive.add(currentFile.getName());
         }
     }
 }
```

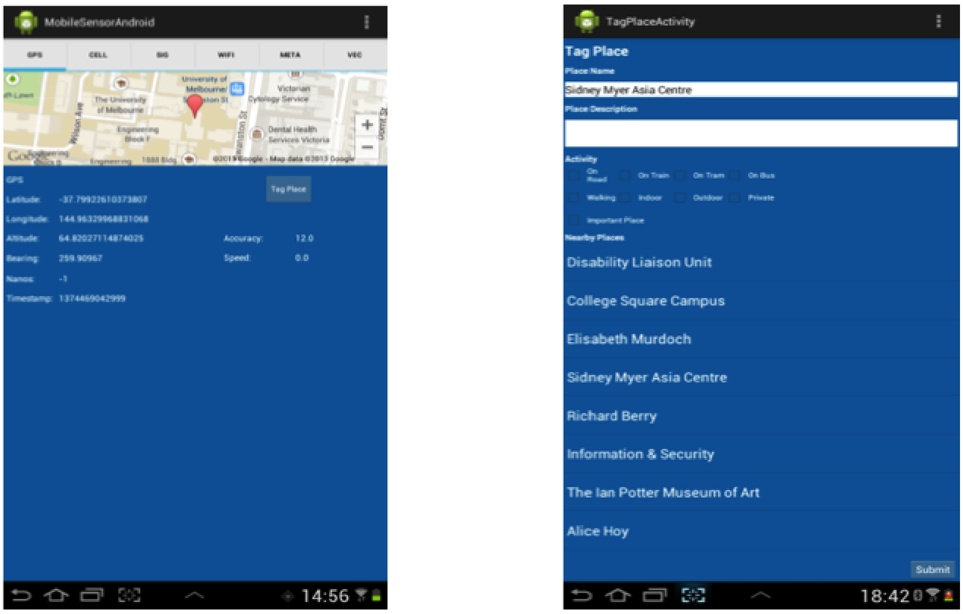
Listing 18: Retrieving Filenames in Android

## A.20 Network Traffic Sensor

The amount of network traffic between cellular and WI-FI networks can be captured via the *TrafficStats* class. Conveniently, the class differentiates between data transferred over cellular networks versus other network connection types.

```
long mobileRxBytes = TrafficStats.getMobileRxBytes();
long mobileRxPackets = TrafficStats.getMobileRxPackets();
long mobileTxBytes = TrafficStats.getMobileTxBytes();
long mobileTxPackets = TrafficStats.getMobileTxPackets();
long totalRxBytes = TrafficStats.getTotalRxBytes();
long totalRxPackets = TrafficStats.getTotalRxPackets();
long totalTxBytes = TrafficStats.getTotalTxBytes();
long totalTxPackets = TrafficStats.getTotalTxPackets();
```

Listing 19: Retrieving Network Traffic Data in Android




(a) Data Being Captured (b) Place Tagger

Figure A.3: MobileSensor User Interface

## A.21 Uploading Data

SQL files are generated by Mobile Sensor containing *INSERT* statements for every entry captured to populate structures detailed in Section A.22. A unique ID field is generated for each entry via a hash function that concatenates the hash of each attribute contained in a row. Each entity, contacts or messages for example are assigned to a separate SQL file. These SQL files are are compressed using tar and GZIP (*tar.gz*) compression. The files are sent to a remote server via a PHP form which will store them and call a job to pipe the statements to a MYSQL client. The UID is set as a unique index so duplicates are not inserted. This process can be run in the background on users' Android devices without them being aware.

## A.22 Relational Data Structure

The ER diagram in Figure A.4 illustrates the structure we used to store data captured from the Mobile Sensor. There is a copy of this database structure on each mobile device using SQLite and a remote copy that syncs with the local databases stored on each device. Each type of information stored on a device such as contacts, messages and call logs are assigned as a separate entity.

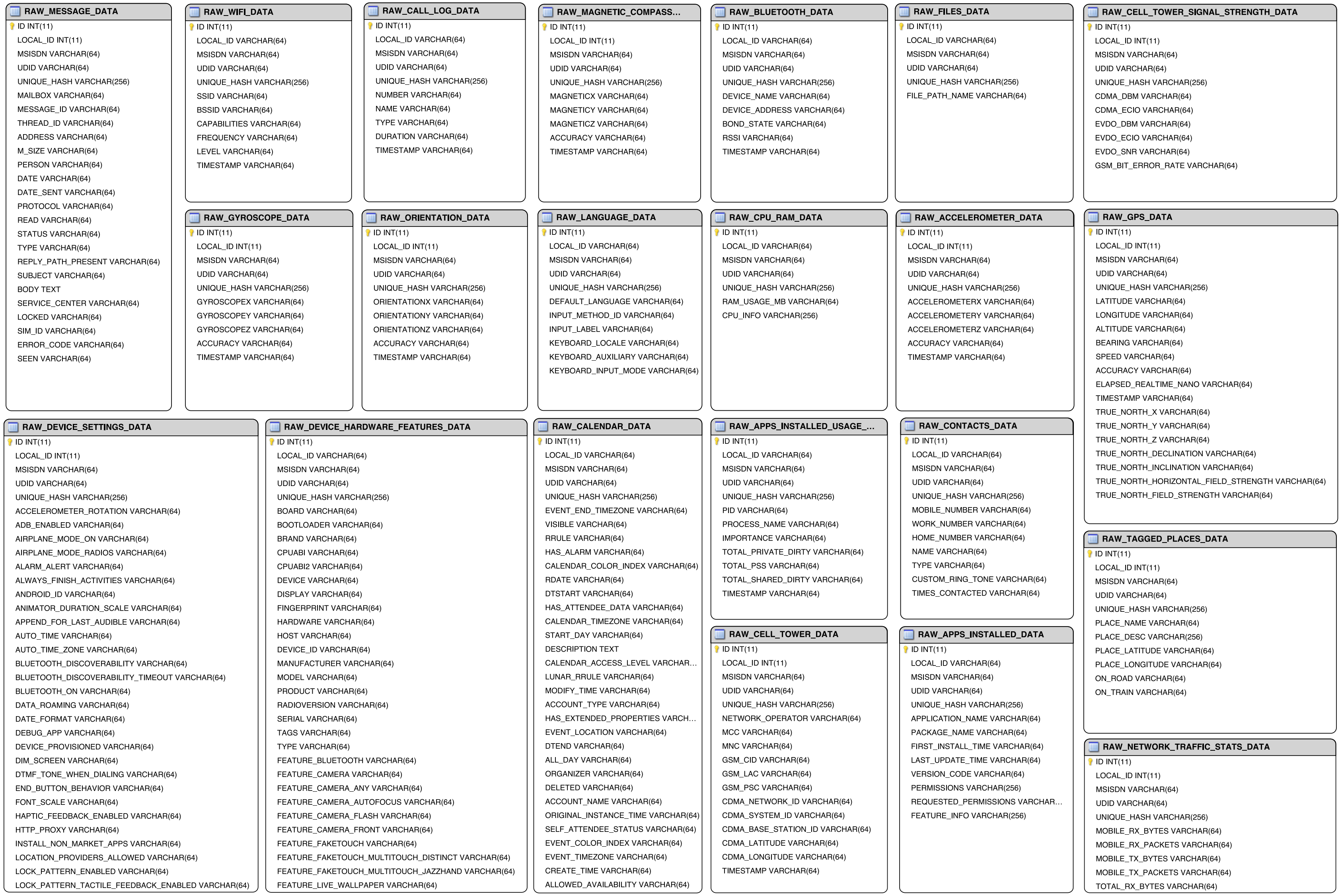



Figure A.4: ER Diagram of our Data Schema to Store Smartphone Data

## A.23 External Data

External data sources that may be useful to supplement with the captured raw internal data include:

- Reverse address/phone listing data
- Australian census
- Business register
- Social media websites
- Cadastre information
- Real estate sale and rental history
- Government public available information (e.g. property titles, births deaths and marriages)

Tools and techniques to infer information will then be applied. Data mining techniques such as decision trees and SOMs may be useful in establishing accurate profiles for smartphone users.

## A.24 Conclusion

Sensitive user data stored on an Andorid device can be easily retrieved using standard API calls. Additional data can also be accessed from API calls provided by externally installed apps such as social media services. The captured data can then be stored in standard Java structures. Mobile data captured can also be efficiently stored in local databases such as SQLite and synchronized with a remote MySQL database.

In this appendix, we demonstrated our approach to capture Android smartphone data.

# Bibliography


[1] Lars Kulik. “Privacy for real-time location-based services”. In: *SIGSPATIAL Special* 1.2 (2009), pp. 9–14.

[2] Roy H. Campbell, Jalal Al-Muhtadi, Prasad Naldurg, Geetanjali Sampemane, and M. Dennis Mickunas. “Towards Security and Privacy for Pervasive Computing”. In: *Proceedings of the Mext-NSF-JSPS International Symposium on Software Security – Theories and Systems (ISSS 2002), Tokyo, Japan, November 8-10, 2002*. Vol. 2609. Lecture Notes in Computer Science. Springer, 2002, pp. 1–15.

[3] Roland Billen, Elsa Joao, and David Forrest. *Dynamic and mobile GIS: investigating changes in space and time*. CRC Press, 2010.

[4] Guanling Chen and Faruq Rahman. “Analyzing Privacy Designs of Mobile Social Networking Applications”. In: *Proceedings of the 2008 IEEE/IPIP International Conference on Embedded and Ubiquitous Computing (EUC 2008), Volume II: Workshops, Shanghai, China, December 17-20, 2008*. IEEE Computer Society, 2008, pp. 83–88.

[5] Aaron Smith. “17% of cell phone owners do most of their online browsing on their phone, rather than a computer or other device”. In: *Washington, DC: Pew Research Center* (2012), pp. 1–16.

[6] Margaret Butler. “Android: Changing the Mobile Landscape”. In: *IEEE Pervasive Computing* 10.1 (2011), pp. 4–7.

[7] Danyl Bosomworth. “Mobile marketing statistics 2015”. In: *Smart Insights site* (2013).

[8] Nathaniel Good, Rachna Dhamija, Jens Grossklags, David Thaw, Steven Aronowitz, Deirdre K. Mulligan, and Joseph A. Konstan. "Stopping spyware at the gate: a user study of privacy, notice and spyware". In: *Proceedings of the 1st Symposium on Usable Privacy and Security (SOUPS 2005), Pittsburgh, Pennsylvania, USA, July 6-8, 2005*. Vol. 93. ACM International Conference Proceeding Series. ACM, 2005, pp. 43–52.

[9] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. "Android permissions demystified". In: *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS 2011), Chicago, Illinois, USA, October 17-21, 2011*. ACM, 2011, pp. 627–638.

[10] A.V. Muratov and R.E. Foley. *Method and system for protecting data within portable electronic devices*. US Patent 7,159,120. 2007.

[11] T. Oh, B. Stackpole, E. Cummins, C. Gonzalez, R. Ramachandran, and Shinyoung Lim. "Best security practices for android, blackberry, and iOS". In: *Proceedings of the 2012 The First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT), Seoul, South Korea, June 18, 2012*. 2012, pp. 42–47.

[12] Lev Grossman and Calif. Cupertino. *Inside Apple CEO Tim Cooks Fight With the FBI*. Ed. by Time. [Online; posted 17-March-2016]. 2016. URL: `http://time.com/4262480/tim-cook-apple-fbi-2/`.

[13] Justin Brickell and Vitaly Shmatikov. "The cost of privacy: destruction of data-mining utility in anonymized data publishing". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008), Las Vegas, Nevada, USA, August 24-27, 2008*. ACM, 2008, pp. 70–78.

[14] Zachary Lutz. *Carrier IQ: What it is, what it isn't, and what you need to know*. Ed. by Engadget.com. [Online; posted 12-December-2011]. 2011. URL: `https://www.engadget.com/2011/12/01/carrier-iq-what-it-is-what-it-isnt-and-what-you-need-to/`.

[15] Matt Apuzzo and Michael S. Schmidt. *Secret Back Door in Some U.S. Phones Sent Data to China, Analysts Say*. Ed. by The New York Times. [Online; posted 15-

November-2016]. 2016. URL: `http://www.nytimes.com/2016/11/16/us/politics/china-phones-software-security.html`.

[16] Michael Barbaro and Tom Zeller. *A Face Is Exposed for AOL Searcher No. 4417749*. Ed. by The New York Times. [Online; posted 09-August-2006]. 2006. URL: `http://www.nytimes.com/2006/08/09/technology/09aol.html`.

[17] Consumerist.com, ed. *AOL User 927 Illuminated*. [Online; posted 07-August-2006]. 2006. URL: `https://consumerist.com/2006/08/07/aol-user-927-illuminated/`.

[18] Arvind Narayanan and Vitaly Shmatikov. "Robust De-anonymization of Large Sparse Datasets". In: *Proceedings of the 2008 IEEE Symposium on Security and Privacy (S&P 2008), Oakland, California, USA, May 18-21, 2008*. IEEE Computer Society, 2008, pp. 111–125.

[19] Anthony Goldbloom. "Data Prediction Competitions – Far More than Just a Bit of Fun". In: *Proceedings of the 10th IEEE International Conference on Data Mining Workshops (ICDMW 2010), Sydney, Australia, December 13, 2010*. IEEE Computer Society, 2010, pp. 1385–1386.

[20] Cynthia Dwork. "Differential Privacy". In: *Proceedings of the 33rd International Colloquium of Automata, Languages and Programming (ICALP 2006), Venice, Italy, July 10-14, 2006*. Vol. 4052. Lecture Notes in Computer Science. Springer, 2006, pp. 1–12.

[21] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), Pisa, Italy, December 15-19, 2008*. IEEE Computer Society, 2008, pp. 413–422.

[22] Samuel D Warren and Louis D Brandeis. "The right to privacy". In: *Harvard law review* (1890), pp. 193–220.

[23] Omer Tene. "Privacy Law's Midlife Crisis: A Critical Assessment of the Second Wave of Global Privacy Laws". In: *Ohio State Law Journal* 74.6 (2013), pp. 1217–1262.

[24] Marc Langheinrich. "Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems". In: *Proceedings of the 3rd International Conference of Ubiquitous Computing (UbiComp 2001), Atlanta, Georgia, USA, September 30 - October 2, 2001*. Vol. 2201. Lecture Notes in Computer Science. Springer, 2001, pp. 273–291.

[25] Glenn Greenwald, Ewen MacAskill, and Laura Poitras. "Edward Snowden: the whistleblower behind the NSA surveillance revelations". In: *The Guardian* 9 (2013).

[26] James B Rule. *Privacy in peril: How we are sacrificing a fundamental right in exchange for security and convenience*. Oxford University Press, 2007.

[27] Alexandre V. Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. "Limiting privacy breaches in privacy preserving data mining". In: *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Diego, CA, USA, June 9-12, 2003*. ACM, 2003, pp. 211–222.

[28] Louise Barkhuus and Anind K. Dey. "Location-Based Services for Mobile Telephony: a Study of Users' Privacy Concerns". In: *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction (INTERACT 2003), Zurich, Switzerland, September 1-5, 2003*. IOS Press, 2003.

[29] Philip E. Agre and Marc Rotenberg. *Technology and Privacy: The New Landscape*. Cambridge, MA, USA: MIT Press Ltd, Sept. 2, 1998. 336 pp.

[30] Larry Hunter. "Public Image". In: *Whole Earth Review* 32 (1985), pp. 32–40.

[31] Helen Nissenbaum. "Protecting Privacy in an Information Age: The Problem of Privacy in Public". English. In: *Law and Philosophy* 17.5–6 (1998), pp. 559–596.

[32] Naresh K. Malhotra, Sung S. Kim, and James Agarwal. "Internet Users' Information Privacy Concerns (IUIPC): The Construct, the Scale, and a Causal Model". In: *Information Systems Research* 15.4 (2004), pp. 336–355.

[33] Latanya Sweeney. "k-Anonymity: A Model for Protecting Privacy". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.5 (2002), pp. 557–570.

[34] Nabil R. Adam and John C. Wortmann. "Security-Control Methods for Statistical Databases: A Comparative Study". In: *ACM Computing Surveys* 21.4 (1989), pp. 515–556.

[35] R. Turn and N. Shapiro. "Privacy and security in databank systems: Measure of effectiveness, costs, and protector-intruder interactions". In: *Proceedings of the Fall Joint Computer Conference, Anaheim, CA, USA, December 5-17, 1972*. ACM. 1972, pp. 49–57.

[36] Francis Y. Chin and Gultekin Ozsoyoglu. "Statistical Database Design". In: *ACM Transactions on Database Systems* 6.1 (1981), pp. 113–139.

[37] I. P. Fellegi. "On the Question of Statistical Confidentiality". In: *Journal of the American Statistical Association* 67.337 (1972), pp. 7–18.

[38] David P. Dobkin, Anita K. Jones, and Richard J. Lipton. "Secure Databases: Protection Against User Influence". In: *ACM Transactions on Database Systems* 4.1 (1979), pp. 97–106.

[39] Schlörer J. "Confidentiality of Statistical Records: A Threat-Monitoring Schelne for On Line Dialogue". In: *Methods Inf Med* 15.1 (1976), pp. 36–52.

[40] Clement T. Yu and Francis Y Chin. "A study on the protection of statistical data bases". In: *Proceedings of the 1977 ACM SIGMOD international conference on Management of data (SIGMOD 1977), Toronto, Ontario, Canada, August 3-5, 1977*. ACM. 1977, pp. 169–181.

[41] Lawrence H Cox. "Suppression methodology and statistical disclosure control". In: *Journal of the American Statistical Association* 75.370 (1980), pp. 377–385.

[42] Steven P. Reiss. "Practical Data-Swapping: The First Steps". In: *ACM Transactions on Database Systems* 9.1 (1984), pp. 20–37.

[43] Chong K. Liew, Uinam J. Choi, and Chung J. Liew. "A Data Distortion by Probability Distribution". In: *ACM Transactions on Database Systems* 10.3 (1985), pp. 395–411.

[44] Dorothy E. Denning. "Secure Statistical Databases with Random Sample Queries". In: *ACM Transactions on Database Systems* 5.3 (1980), pp. 291–315.

[45] Leland L. Beck. "A Security Mechanism for Statistical Databases". In: *ACM Transactions on Database Systems* 5.3 (1980), pp. 316–338.

[46] I. P. Fellegi and J. J. Phillips. "Statistical Confidentiality: Some Theory and Application to Data Dissemination". In: *Annals of Economic and Social Measurement, Volume 3, number 2*. National Bureau of Economic Research, Inc, 1974, pp. 399–409.

[47] Mohammed Inam Ul Haq. "Insuring individual's privacy from statistical data base users". In: *Proceedings of the American Federation of Information Processing Societies: 1975 National Computer Conference, Anaheim, CA, USA, May 19-22, 1975*. Vol. 44. AFIPS Conference Proceedings. AFIPS Press, 1975, pp. 941–946.

[48] Mohammad Inam ul Haq. "On Safeguarding Statistical Disclosure by Giving Approximate Answers to Queries". In: *Proceedings of the International Computing Symposium, Liège, Belgium, April 4-7, 1977*. North-Holland, 1977, pp. 491–495.

[49] James Achugbue and Francis Chin. "The Effectiveness Of Output Modification By Rounding For Protection Of Statistical Data Bases". In: *INFOR: Information Systems and Operational Research* 17.3 (1979), pp. 209–218.

[50] T Dalenius. "A simple procedure for controlled rounding". In: *Statistik Tidskrift* 3 (1981), pp. 202–208.

[51] Tore Dalenius. "Towards a methodology for statistical disclosure control". In: *statistik Tidskrift* 15.429-444 (1977), pp. 2–1.

[52] Latanya Sweeney. "Weaving Technology and Policy Together to Maintain Confidentiality". In: *The Journal of Law, Medicine & Ethics* 25.2-3 (1997), pp. 98–110.

[53] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A. Stephan, Stanley F. Nelson, and David W. Craig. "Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures Using High-Density SNP Genotyping Microarrays". In: *PLoS Genet* 4.8 (Aug. 2008), pp. 1–9.

[54] George Church, Catherine Heeney, Naomi Hawkins, Jantina de Vries, Paula Boddington, Jane Kaye, Martin Bobrow, Bruce Weir, and P3G Consortium. "Public

Access to Genome-Wide Data: Five Views on Balancing Research with Privacy and Protection". In: *PLoS Genet* 5.10 (Oct. 2009), pp. 1–4.

[55] Tore Daleniusl. "Finding a Needle In a Haystack or Identifying Anonymous Census Records". In: *Journal of Official Statistics* 2.3 (1986), pp. 329–336.

[56] Vijay S. Iyengar. "Transforming data to satisfy privacy constraints". In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002), Edmonton, Alberta, Canada, July 23-26, 2002*. ACM, 2002, pp. 279–288.

[57] Leon Willenborg and Ton De Waal. *Statistical disclosure control in practice*. 111. Springer Science & Business Media, 1996.

[58] Adam Meyerson and Ryan Williams. "On the Complexity of Optimal K-Anonymity". In: *Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2004), Paris, France, June 14-16, 2004*. ACM, 2004, pp. 223–228.

[59] Pierangela Samarati. "Protecting Respondents' Identities in Microdata Release". In: *IEEE Transactions on Knowledge and Data Engineering* 13.6 (2001), pp. 1010–1027.

[60] Latanya Sweeney. "Achieving k-Anonymity Privacy Protection Using Generalization and Suppression". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.5 (2002), pp. 571–588.

[61] Latanya Sweeney. "Datafly: A System for Providing Anonymity in Medical Data". In: *Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Security XI: Status and Prospects, Lake Tahoe, California, USA, August 10-13, 1997*. Vol. 113. IFIP Conference Proceedings. Chapman & Hall, 1997, pp. 356–381.

[62] William Winkler. *Using simulated annealing for k-anonymity*. Tech. rep. Research Report 2002-07, US Census Bureau Statistical Research Division, 2002.

[63] Charu C. Aggarwal. "On k-Anonymity and the Curse of Dimensionality". In: *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB*

*2005), Trondheim, Norway, August 30 - September 2, 2005*. ACM, 2005, pp. 901–909.

[64] Roberto J. Bayardo Jr. and Rakesh Agrawal. "Data Privacy through Optimal k-Anonymization". In: *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005), Tokyo, Japan, April 5-8, 2005*. IEEE Computer Society, 2005, pp. 217–228.

[65] Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. "Approximation algorithms for k-anonymity". In: *Journal of Privacy Technology (JOPT)* (2005).

[66] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. "*L*-diversity: Privacy beyond *k*-anonymity". In: *TKDD* 1.1 (2007).

[67] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity". In: *Proceedings of the 23rd International Conference on Data Engineering (ICDE 2007), The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*. IEEE Computer Society, 2007, pp. 106–115.

[68] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. "The Earth Mover's Distance as a Metric for Image Retrieval". In: *International Journal of Computer Vision* 40.2 (2000), pp. 99–121.

[69] Charu C. Aggarwal and Philip S. Yu. "A General Survey of Privacy-Preserving Data Mining Models and Algorithms". In: *Privacy-Preserving Data Mining - Models and Algorithms*. Vol. 34. Advances in Database Systems. Springer, 2008, pp. 11–52.

[70] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Proceedings of the Theory of Cryptography, Third Theory of Cryptography Conference (TCC 2006), New York, NY, USA, March 4-7, 2006*. Vol. 3876. Lecture Notes in Computer Science. Springer, 2006, pp. 265–284.

[71] Frank McSherry and Kunal Talwar. "Mechanism Design via Differential Privacy". In: *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer*

*Science (FOCS 2007), Providence, RI, USA, October 20-23, 2007*. IEEE Computer Society, 2007, pp. 94–103.

[72] Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. "Protecting Privacy Against Location-Based Personal Identification". In: *Proceedings of the Secure Data Management, Second VLDB Workshop (SDM 2005), Trondheim, Norway, September 2-3, 2005*. Vol. 3674. Lecture Notes in Computer Science. Springer, 2005, pp. 185–199.

[73] Matt Duckham and Lars Kulik. "Location privacy and location-aware computing". In: *Dynamic & mobile GIS: investigating change in space and time* 3 (2006), pp. 35–51.

[74] Matt Duckham and Lars Kulik. "A Formal Model of Obfuscation and Negotiation for Location Privacy". In: *Proceedings of Third International Conference, PERVASIVE 2005, Munich, Germany, May 8-13, 2005*. Vol. 3468. Lecture Notes in Computer Science. Springer, 2005, pp. 152–170.

[75] Matt Duckham and Lars Kulik. "Simulation of Obfuscation and Negotiation for Location Privacy". In: *Proceedings of the Spatial Information Theory, International Conference (COSIT 2005), Ellicottville, NY, USA, September 14-18, 2005*. Vol. 3693. Lecture Notes in Computer Science. Springer, 2005, pp. 31–48.

[76] Andreas Pfitzmann and Marit Köhntopp. "Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology". In: *Proceedings of the Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000*. Lecture Notes in Computer Science. Springer, 2000, pp. 1–9.

[77] Marco Gruteser and Baik Hoh. "On the Anonymity of Periodic Location Samples". In: *Proceedings of the Security in Pervasive Computing, Second International Conference (SPC 2005), Boppard, Germany, April 6-8, 2005*. Vol. 3450. Lecture Notes in Computer Science. Springer, 2005, pp. 179–192.

[78] Samuel S Blackman. *Multiple-target tracking with radar applications*. 1986.

[79] Baik Hoh and Marco Gruteser. "Protecting Location Privacy Through Path Confusion". In: *Proceedings of the First International Conference on Security and Privacy*

*for Emerging Areas in Communications Networks (SecureComm 2005), Athens, Greece, 5-9 September, 2005*. IEEE, 2005, pp. 194–205.

[80] Chris Y. T. Ma, David K. Y. Yau, Nung Kwan Yip, and Nageswara S. V. Rao. "Privacy Vulnerability of Published Anonymous Mobility Traces". In: *IEEE/ACM Transactions on Networking* 21.3 (2013), pp. 720–733.

[81] Muyuan Li, Haojin Zhu, Zhaoyu Gao, Si Chen, Le Yu, Shangqian Hu, and Kui Ren. "All your location are belong to us: breaking mobile social networks for automated user location tracking". In: *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2014), Philadelphia, PA, USA, August 11-14, 2014*. ACM, 2014, pp. 43–52.

[82] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. "De-anonymization attack on geolocated data". In: *Journal of Computer and System Sciences* 80.8 (2014), pp. 1597–1614.

[83] Yuan Zhong, Nicholas Jing Yuan, Wen Zhong, Fuzheng Zhang, and Xing Xie. "You Are Where You Go: Inferring Demographic Attributes from Location Check-ins". In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM 2015), Shanghai, China, February 2-6, 2015*. ACM, 2015, pp. 295–304.

[84] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. "Unique in the crowd: The privacy bounds of human mobility". In: *Scientific Reports* 3 (2013).

[85] Natalia Marmasse and Chris Schmandt. "Location-Aware Information Delivery with *ComMotion*". In: *Proceedings of the Handheld and Ubiquitous Computing, Second International Symposium (HUC 2000), Bristol, UK, September 25-27, 2000*. Vol. 1927. Lecture Notes in Computer Science. Springer, 2000, pp. 157–171.

[86] Natalia Marmasse. "Providing lightweight telepresence in mobile communication to enhance collaborative living". PhD thesis. Massachusetts Institute of Technology (MIT). Dept. of Architecture. Program in Media Arts and Sciences, 2004.

[87] John Krumm. “Inference Attacks on Location Tracks”. In: *Proceedings of the 5th International Conference, PERVASIVE 2007, Toronto, Canada, May 13-16, 2007*. Vol. 4480. Lecture Notes in Computer Science. Springer, 2007, pp. 127–143.

[88] Philippe Golle and Kurt Partridge. “On the Anonymity of Home/Work Location Pairs”. In: *Proceedings of the 7th International Conference on Pervasive Computing (Pervasive 2009), Nara, Japan, May 11-14, 2009*. Vol. 5538. Lecture Notes in Computer Science. Springer, 2009, pp. 390–397.

[89] Hui Zang and Jean Bolot. “Anonymization of location data does not work: a large-scale measurement study”. In: *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking (MOBICOM 2011), Las Vegas, Nevada, USA, September 19-23, 2011*. ACM, 2011, pp. 145–156.

[90] Daniel Ashbrook and Thad Starner. “Using GPS to learn significant locations and predict movement across multiple users”. In: *Personal and Ubiquitous Computing* 7.5 (2003), pp. 275–286.

[91] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. “Extracting places from traces of locations”. In: *Proceedings of the 2nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH 2004), Philadelphia, PA, USA, October 1, 2004*. ACM, 2004, pp. 110–118.

[92] Ramaswamy Hariharan and Kentaro Toyama. “Project Lachesis: Parsing and Modeling Location Histories”. In: *Proceedings of the Third International Conference Geographic Information Science (GIScience 2004), Adelphi, MD, USA, October 20-23, 2004*. Vol. 3234. Lecture Notes in Computer Science. Springer, 2004, pp. 106–124.

[93] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. “Enhancing Security and Privacy in Traffic-Monitoring Systems”. In: *IEEE Pervasive Computing* 5.4 (2006), pp. 38–46.

[94] Donald J. Patterson, Lin Liao, Dieter Fox, and Henry A. Kautz. “Inferring High-Level Behavior from Low-Level Sensors”. In: *Proceedings of the 5th International Conference of Ubiquitous Computing (UbiComp 2003), Seattle, WA, USA, October*

*12-15, 2003*. Vol. 2864. Lecture Notes in Computer Science. Springer, 2003, pp. 73–89.

[95] Jon Froehlich and John Krumm. *Route prediction from trip observations*. Tech. rep. SAE Technical Paper, 2008.

[96] John Krumm. *A markov model for driver turn prediction*. Tech. rep. SAE Technical Paper, 2008.

[97] Luca Canzian and Mirco Musolesi. "Trajectories of depression: unobtrusive monitoring of depressive states by means of smartphone mobility traces analysis". In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2015), Osaka, Japan, September 7-11, 2015*. ACM, 2015, pp. 1293–1304.

[98] Reynold Cheng, Yu Zhang, Elisa Bertino, and Sunil Prabhakar. "Preserving User Location Privacy in Mobile Data Management Infrastructures". In: *Proceedings of the Privacy Enhancing Technologies, 6th International Workshop (PET 2006), Cambridge, UK, June 28-30, 2006*. Vol. 4258. Lecture Notes in Computer Science. Springer, 2006, pp. 393–412.

[99] Emre Kaplan, Thomas Brochmann Pedersen, Erkay Savas, and Yücel Saygin. "Discovering private trajectories using background information". In: *Data & Knowledge Engineering* 69.7 (2010), pp. 723–736.

[100] Chi-Yin Chow, Mohamed F. Mokbel, Joe Naps, and Suman Nath. "Approximate Evaluation of Range Nearest Neighbor Queries with Quality Guarantee". In: *Proceedings of the 11th International Symposium on Spatial and Temporal Databases (SSTD 2009), Aalborg, Denmark, July 8-10, 2009*. Vol. 5644. Lecture Notes in Computer Science. Springer, 2009, pp. 283–301.

[101] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. "The New Casper: Query Processing for Location Services without Compromising Privacy". In: *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB '06), Seoul, Korea, September 12-15, 2006*. ACM, 2006, pp. 763–774.

[102] Gabriel Ghinita, Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. "Preventing velocity-based linkage attacks in location-aware applications". In: *Proceed-*

*ings of the 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (ACM-GIS 2009), Seattle, Washington, USA, November 4-6, 2009*. ACM, 2009, pp. 246–255.

[103] Amina Hossain, Anthony Quattrone, Egemen Tanin, and Lars Kulik. "On the Effectiveness of Removing Location Information from Trajectory Data for Preserving Location Privacy". In: *Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS 2016), San Francisco Bay Area, California, USA, October 31, 2015*. 2016.

[104] Alastair R. Beresford and Frank Stajano. "Location Privacy in Pervasive Computing". In: *IEEE Pervasive Computing* 2.1 (2003), pp. 46–55.

[105] Alastair R. Beresford and Frank Stajano. "Mix Zones: User Privacy in Location-aware Services". In: *Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2004 Workshops), Orlando, FL, USA, March 14-17, 2004*. IEEE Computer Society, 2004, pp. 127–131.

[106] Julien Freudiger, Reza Shokri, and Jean-Pierre Hubaux. "On the Optimal Placement of Mix Zones". In: *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies (PETS 2009), Seattle, WA, USA, August 5-7, 2009*. Vol. 5672. Lecture Notes in Computer Science. Springer, 2009, pp. 216–234.

[107] Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. "An anonymous communication technique using dummies for location-based services". In: *Proceedings of the International Conference on Pervasive Services 2005 (ICPS '05), Santorini, Greece, July 11-14, 2005*. IEEE Computer Society, 2005, pp. 88–97.

[108] Man Lung Yiu, Christian S. Jensen, Xuegang Huang, and Hua Lu. "SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services". In: *Proceedings of the 24th International Conference on Data Engineering (ICDE 2008), Cancún, México, April 7-12, 2008*. IEEE Computer Society, 2008, pp. 366–375.

[109] Man Lung Yiu, Christian S. Jensen, Jesper Møller, and Hua Lu. "Design and analysis of a ranking approach to private location-based services". In: *ACM Transactions on Database Systems* 36.2 (2011), pp. 1–42.

[110] Hua Lu, Christian S. Jensen, and Man Lung Yiu. "PAD: privacy-area aware, dummy-based location privacy in mobile services". In: *Proceedings of the Seventh ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE 2008), Vancouver, British Columbia, Canada, June 13, 2008*. ACM, 2008, pp. 16–23.

[111] P. Wightman, W. Coronell, D. Jabba, M. Jimeno, and M. Labrador. "Evaluation of Location Obfuscation techniques for privacy in location based information systems". In: *Proceedings of the 2011 IEEE Third Latin-American Conference on Communications (LATINCOM 2011), Belem, Brazil, October 24-26, 2011*. 2011, pp. 1–6.

[112] Miguel E. Andrés, Nicolás Emilio Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. "Geo-indistinguishability: differential privacy for location-based systems". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS 2013), Berlin, Germany, November 4-8, 2013*. ACM, 2013, pp. 901–914.

[113] Gabriel Ghinita, Keliang Zhao, Dimitris Papadias, and Panos Kalnis. "A reciprocal framework for spatial K-anonymity". In: *Information Systems* 35.3 (2010), pp. 299–314.

[114] Sergio Mascetti and Claudio Bettini. "A Comparison of Spatial Generalization Algorithms for LBS Privacy Preservation". In: *Proceedings of the 8th International Conference on Mobile Data Management (MDM 2007), Mannheim, Germany, May 7-11, 2007*. IEEE, 2007, pp. 258–262.

[115] Marco Gruteser and Dirk Grunwald. "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking". In: *Proceedings of the First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003), San Francisco, CA, USA, May 5-8, 2003*. USENIX, 2003.

[116] Marco Gruteser and Xuan Liu. "Protecting Privacy in Continuous Location-Tracking Applications". In: *IEEE Security & Privacy* 2.2 (2004), pp. 28–34.

[117] Bugra Gedik and Ling Liu. “Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms”. In: *IEEE Transactions on Mobile Computing* 7.1 (2008), pp. 1–18.

[118] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitri Papadias. “Preserving anonymity in location based services”. In: *Computer Science Department, National University of Singapore, Singapore, Technical* (2006).

[119] Tanzima Hashem, Mohammed Eunus Ali, Lars Kulik, Egemen Tanin, and Anthony Quattrone. “Protecting privacy for group nearest neighbor queries with crowd-sourced data and computing”. In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2013), Zurich, Switzerland, September 8-12, 2013*. ACM, 2013, pp. 559–562.

[120] Tanzima Hashem and Lars Kulik. “Safeguarding Location Privacy in Wireless Ad-Hoc Networks”. In: *Proceedings of the 9th International Conference of Ubiquitous Computing (UbiComp 2007), Innsbruck, Austria, September 16-19, 2007*. Vol. 4717. Lecture Notes in Computer Science. Springer, 2007, pp. 372–390.

[121] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. “PRIVE: anonymous location-based queries in distributed mobile systems”. In: *Proceedings of the 16th International Conference on World Wide Web (WWW 2007), Banff, Alberta, Canada, May 8-12, 2007*. ACM, 2007, pp. 371–380.

[122] Nishkam Ravi, Marco Gruteser, and Liviu Iftode. “Non-Inference: An Information Flow Control Model for Location-based Services”. In: *Proceedings of the 3rd Annual International ICST Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MOBIQUITOUS 2006), San Jose, California, USA, July 17-21, 2006*. IEEE Computer Society, 2006, pp. 1–10.

[123] John Canny. “Some techniques for privacy in Ubicomp and context-aware applications”. In: *Proceedings of the Workshop on Socially Informed Design of Privacy-enhancing Solutions in Ubiquitous Computing, Seattle, WA, October, 2003*. Citeseer. 2002.

[124] Ali Khoshgozaran and Cyrus Shahabi. “Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy”. In: *Proceed-*

*ings of the 10th International Symposium on Advances in Spatial and Temporal Databases (SSTD 2007), Boston, MA, USA, July 16-18, 2007*. Vol. 4605. Lecture Notes in Computer Science. Springer, 2007, pp. 239–257.

[125] Ge Zhong, Ian Goldberg, and Urs Hengartner. "Louis, lester and pierre: Three protocols for location privacy". In: *Proceedings of the International Workshop on Privacy Enhancing Technologies (PET 2007), Ottawam Canada, June 20-22, 2007*. Springer. 2007, pp. 62–76.

[126] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. "Private queries in location based services: anonymizers are not necessary". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2008), Vancouver, BC, Canada, June 10-12, 2008*. ACM, 2008, pp. 121–132.

[127] Jason Cornwell, Ian Fette, Gary Hsieh, Madhu K. Prabaker, Jinghai Rao, Karen P. Tang, Kami Vaniea, Lujo Bauer, Lorrie Faith Cranor, Jason I. Hong, Bruce McLaren, Mike Reiter, and Norman M. Sadeh. "User-Controllable Security and Privacy for Pervasive Computing". In: *Proceedings of the Eighth IEEE Workshop on Mobile Computing Systems and Applications (HotMobile 2007), Tucson, Arizona, USA, March 8-9, 2007*. IEEE Computer Society, 2007, pp. 14–19.

[128] Ponnurangam Kumaraguru and Lorrie Faith Cranor. *Privacy indexes: A Survey of Westin's Studies*. Tech. rep. Institute for Software Research, Carnegie Mellon University, 2005.

[129] Judith S. Olson, Jonathan Grudin, and Eric Horvitz. "A study of preferences for sharing and privacy". In: *Proceedings of the 2005 Conference on Human Factors in Computing Systems (CHI 2005), Portland, Oregon, USA, April 2-7, 2005*. ACM, 2005, pp. 1985–1988.

[130] Tiancheng Li and Ninghui Li. "On the tradeoff between privacy and utility in data publishing". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009), Paris, France, June 28 - July 1, 2009*. ACM, 2009, pp. 517–526.

[131] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. “Preserving privacy in GPS traces via uncertainty-aware path cloaking”. In: *Proceedings of the 2007 ACM Conference on Computer and Communications Security (CCS 2007), Alexandria, Virginia, USA, October 28-31, 2007*. ACM, 2007, pp. 161–171.

[132] Reza Shokri. “Quantifying and protecting location privacy”. In: *it - Information Technology* 57.4 (2015), pp. 257–263.

[133] Eija Kaasinen. “User needs for location-aware mobile services”. In: *Personal and Ubiquitous Computing* 7.1 (2003), pp. 70–79.

[134] Louise Barkhuus. “Privacy in Location-Based Services, Concern vs. Coolness”. In: *Proceedings of the MobileHCI Workshop on Location System Privacy and Control, Glasgow, UK, September 13-16, 2004*. Vol. 4. 2004, pp. 24–29.

[135] Giovanni Iachello, Ian E. Smith, Sunny Consolvo, Gregory D. Abowd, Jeff Hughes, James Howard, Fred Potter, James Scott, Timothy Sohn, Jeffrey Hightower, and Anthony LaMarca. “Control, Deception, and Communication: Evaluating the Deployment of a Location-Enhanced Messaging Service”. In: *Proceedings of the 7th International Conference of Ubiquitous Computing (UbiComp 2005), Tokyo, Japan, September 11-14, 2005*. Vol. 3660. Lecture Notes in Computer Science. Springer, 2005, pp. 213–231.

[136] Martin Colbert. “A diary study of rendezvousing: implications for position-aware computing and communications for the general public”. In: *Proceedings of the ACM 2001 International Conference on Supporting Group Work (GROUP 2001), Boulder, Colorado, USA, September 30 - October 3, 2001*. ACM, 2001, pp. 15–23.

[137] George Danezis, Stephen Lewis, and Ross J. Anderson. “How Much Is Location Privacy Worth?” In: *Proceedings of the 4th Annual Workshop on the Economics of Information Security (WEIS 2005), Harvard University, Cambridge, MA, USA, June 1-3, 2005*. 2005.

[138] Daniel Cvrcek, Marek Kumpost, Vashek Matyas, and George Danezis. “A study on the value of location privacy”. In: *Proceedings of the 2006 ACM Workshop on Privacy in the Electronic Society (WPES 2006), Alexandria, VA, USA, October 30, 2006*. ACM, 2006, pp. 109–118.

[139] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otávio Alvares. “A clustering-based approach for discovering interesting places in trajectories”. In: *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC 2008), Fortaleza, Ceara, Brazil, March 16-20, 2008*. ACM, 2008, pp. 863–868.

[140] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. “Mining interesting locations and travel sequences from GPS trajectories”. In: *Proceedings of the 18th International Conference on World Wide Web, (WWW2009), Madrid, Spain, April 20-24, 2009*. ACM, 2009, pp. 791–800.

[141] Tun-Hao You, Wen-Chih Peng, and Wang-Chien Lee. “Protecting Moving Trajectories with Dummies”. In: *Proceedings of the 8th International Conference on Mobile Data Management, (MDM 2007), Mannheim, Germany, May 7-11, 2007*. 2007, pp. 278–282.

[142] Manolis Terrovitis and Nikos Mamoulis. “Privacy Preservation in the Publication of Trajectories”. In: *Proceedings of the Ninth International Conference on Mobile Data Management (MDM 2008), Beijing, China, April 27-30, 2008*. IEEE, 2008, pp. 65–72.

[143] Anna Monreale, Roberto Trasarti, Dino Pedreschi, Chiara Renso, and Vania Bogorny. “C-safety: a framework for the anonymization of semantic trajectories”. In: *Transactions on Data Privacy* 4.2 (2011), pp. 73–101.

[144] Osman Abul, Francesco Bonchi, and Mirco Nanni. “Never Walk Alone: Uncertainty for Anonymity in Moving Objects Databases”. In: *Proceedings of the 24th International Conference on Data Engineering (ICDE 2008), Cancún, México, April 7-12, 2008*. Ed. by Gustavo Alonso, José A. Blakeley, and Arbee L. P. Chen. IEEE Computer Society, 2008, pp. 376–385.

[145] Roman Yarovoy, Francesco Bonchi, Laks V. S. Lakshmanan, and Wendy Hui Wang. “Anonymizing moving objects: how to hide a MOB in a crowd?” In: *Proceedings of the 12th International Conference on Extending Database Technology (EDBT 2009), Saint Petersburg, Russia, March 24-26, 2009*. Vol. 360. ACM International Conference Proceeding Series. ACM, 2009, pp. 72–83.

[146] Zheng Huo, Yi Huang, and Xiaofeng Meng. "History Trajectory Privacy-preserving Through Graph Partition". In: *Proceedings of the 1st International Workshop on Mobile Location-based Service (MLBS 2011), Beijing, China, September 17-21, 2011*. MLBS '11. ACM, 2011, pp. 71–78.

[147] Zheng Huo, Xiaofeng Meng, Haibo Hu, and Yi Huang. "You Can Walk Alone: Trajectory Privacy-Preserving through Significant Stays Protection". In: *Proceedings of the 17th International Conference on Database Systems for Advanced Applications (DASFAA 2012), Busan, South Korea, April 15-18 2012*. Vol. 7238. Lecture Notes in Computer Science. Springer, 2012, pp. 351–366.

[148] Wenliang Du, Yunghsiang S. Han, and Shigang Chen. "Privacy-Preserving Multivariate Statistical Analysis: Linear Regression and Classification". In: *Proceedings of the Fourth SIAM International Conference on Data Mining (SDM 2004), Lake Buena Vista, Florida, USA, April 22-24, 2004*. SIAM, 2004, pp. 222–233.

[149] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[150] Tingting Chen and Sheng Zhong. "Privacy-Preserving Backpropagation Neural Network Learning". In: *IEEE Transactions on Neural Networks* 20.10 (2009), pp. 1554–1564.

[151] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". English. In: *Machine Learning* 20.3 (1995), pp. 273–297.

[152] Hwanjo Yu, Xiaoqian Jiang, and Jaideep Vaidya. "Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data". In: *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006), Dijon, France, April 23-27, 2006*. ACM, 2006, pp. 603–610.

[153] Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. "Privacy-Preserving SVM Classification on Vertically Partitioned Data". In: *Proceedings of the 10th Pacific-Asia Conference of Advances in Knowledge Discovery and Data Mining (PAKDD 2006), Singapore, April 9-12, 2006*. Vol. 3918. Lecture Notes in Computer Science. Springer, 2006, pp. 647–656.

[154] Mark Levene. “Rokach Lior and Maimon Oded: *Data Mining with Decision Trees: Theory and Applications* World Scientific (2008) ISBN-13 978-981-277-171-1”. In: *The Computer Journal* 53.4 (2010), p. 489.

[155] Fatih Emekçi, Ozgur D. Sahin, Divyakant Agrawal, and Amr El Abbadi. “Privacy preserving decision tree learning over multiple parties”. In: *Data & Knowledge Engineering* 63.2 (2007), pp. 348–361.

[156] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32.

[157] Santu Rana, Sunil Kumar Gupta, and Svetha Venkatesh. “Differentially Private Random Forest with High Utility”. In: *Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM 2015), Atlantic City, NJ, USA, November 14-17, 2015*. IEEE Computer Society, 2015, pp. 955–960.

[158] Shuguo Han and Wee Keong Ng. “Privacy-Preserving Self-Organizing Map”. In: *Proceedings of the 9th International Conference of Data Warehousing and Knowledge Discovery (DaWaK 2007), Regensburg, Germany, September 3-7, 2007*. Vol. 4654. Lecture Notes in Computer Science. Springer, 2007, pp. 428–437.

[159] Dong-Her Shih, Binshan Lin, Hsiu-Sen Chiang, and Ming-Hung Shih. “Security aspects of mobile phone virus: a critical survey”. In: *Industrial Management and Data Systems* 108.4 (2008), pp. 478–494.

[160] A Schmidt and Sahin Albayrak. *Malicious software for smartphones*. Tech. rep. Technische Universität Berlin, DAI-Labor, Tech. Rep. TUB-DAI, 2008, pp. 08–01.

[161] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. “A survey of mobile malware in the wild”. In: *Proceedings of the 1st ACM Workshop Security and Privacy in Smartphones and Mobile Devices (SPSM 2011), Co-located with CCS 2011, Chicago, IL, USA, October 17, 2011*. ACM, 2011, pp. 3–14.

[162] Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra. “A Survey on Security for Mobile Devices”. In: *IEEE Communications Surveys & Tutorials* 15.1 (2013), pp. 446–471.

[163] Sancheng Peng, Shui Yu, and Aimin Yang. "Smartphone Malware and Its Propagation Modeling: A Survey". In: *IEEE Communications Surveys and Tutorials* 16.2 (2014), pp. 925–941.

[164] Billy Lau, Yeongjin Jang, Chengyu Song, Tielei Wang, PH Chung, and P Royal. "Mactans: Injecting malware into iOS devices via malicious chargers". In: *Proceedings of Black Hat USA* (2013).

[165] Kaspersky Lab, ed. *The very first mobile malware: how Kaspersky Lab discovered Cabir*. [Online; posted 16-June-2014]. 2014. URL: `http://www.kaspersky.co.in/about/news/virus/2014/The-very-first-mobile-malware-how-Kaspersky-Lab-discovered-Cabir`.

[166] Kaspersky Lab, ed. *Antivirus Protection & Internet Security Software*. [Online; posted 09-August-2010]. 2010. URL: `http://me.kaspersky.com/en/about/news/virus/2010/First_SMS_Trojan_detected_for_smartphones_running_Android`.

[167] F-Secure Corporation, ed. *WORM:IPHONEOS/IKEE*. [Online; posted 25-November-2009]. 2009. URL: `https://www.f-secure.com/v-descs/worm_iphoneos_ikee.shtml`.

[168] Juniper Networks, ed. *2011 Mobile Threats Report*. [Online; posted February-2011]. 2011. URL: `http://www.juniper.net/us/en/local/pdf/additional-resources/jnpr-2011-mobile-threats-report.pdf`.

[169] Mikko Hypponen and Jarno Niemela. *BRADOR*. Ed. by F-Secure Corporation. [Online; posted 06-August-2004]. 2004. URL: `https://www.f-secure.com/v-descs/brador.shtml`.

[170] Manuel Egele, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. "PiOS: Detecting Privacy Leaks in iOS Applications". In: *Proceedings of the Network and Distributed System Security Symposium (NDSS 2011), San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society, 2011.

[171] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. "AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on

a Large Scale”. In: *Proceedings of the 5th International Conference of Trust and Trustworthy Computing (TRUST 2012), Vienna, Austria, June 13-15, 2012*. Vol. 7344. Lecture Notes in Computer Science. Springer-Verlag GmbH, June 5, 2012, pp. 291–307.

[172] John Paul Dunning. “Taming the Blue Beast: A Survey of Bluetooth Based Threats”. In: *IEEE Security & Privacy* 8.2 (2010), pp. 20–27.

[173] Erika Chin, Adrienne Porter Felt, Vyas Sekar, and David Wagner. “Measuring user confidence in smartphone security and privacy”. In: *Proceedings of the Symposium On Usable Privacy and Security (SOUPS ’12), Washington, DC, USA - July 11 - 13, 2012*. ACM, 2012, p. 1.

[174] Adrienne Porter Felt, Serge Egelman, and David Wagner. “I’Ve Got 99 Problems, but Vibration Ain’T One: A Survey of Smartphone Users’ Concerns”. In: *Proceedings of the 2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM 2012), Raleigh, NC, USA, October 19, 2012*. SPSM ’12. Raleigh, North Carolina, USA: ACM, 2012, pp. 33–44.

[175] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. “Android permissions: user attention, comprehension, and behavior”. In: *Proceedings of the Symposium On Usable Privacy and Security (SOUPS ’12), Washington, DC, USA - July 11 - 13, 2012*. ACM, 2012, p. 3.

[176] Alexios Mylonas, Anastasia Kastania, and Dimitris Gritzalis. “Delegate the smartphone user? Security awareness in smartphone platforms”. In: *Computers & Security* 34 (2013), pp. 47–66.

[177] Serge Egelman, Adrienne Porter Felt, and David Wagner. “Choice Architecture and Smartphone Privacy: There’s a Price for That”. In: *The Economics of Information Security and Privacy*. Springer, 2013, pp. 211–236.

[178] The United Nations. *Universal Declaration of Human Rights*. Office of the United Nations High Commissioner for Human Rights, 1948.

[179] Richard A. Becker, Ramón Cáceres, Karrie Hanson, Sibren Isaacman, Ji Meng Loh, Margaret Martonosi, James Rowland, Simon Urbanek, Alexander Varshavsky, and

Chris Volinsky. “Human mobility characterization from cellular network data”. In: *Communications of the ACM* 56.1 (2013), pp. 74–82.

[180] Miao Lin, Wen-Jing Hsu, and Zhuo Qi Lee. “Predictability of individuals’ mobility with high-resolution positioning data”. In: *Proceedings of the 14th ACM International Conference on Ubiquitous Computing (UbiComp 2012) Pittsburgh, PA, USA, September 5-8, 2012*. ACM, 2012, pp. 381–390.

[181] Maria Luisa Damiani, Elisa Bertino, and Claudio Silvestri. “Protecting location privacy against spatial inferences: the PROBE approach”. In: *Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS (SPRINGL 2009), Seattle, WA, USA, November 3, 2009*. ACM, 2009, pp. 32–41.

[182] Tanzima Hashem and Lars Kulik. “”Don’t trust anyone”: Privacy protection for location-based services”. In: *Pervasive and Mobile Computing* 7.1 (2011), pp. 44–59.

[183] Marius Wernke, Frank Dürr, and Kurt Rothermel. “PShare: Position sharing for location privacy based on multi-secret sharing”. In: *Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communications (PerCom 2012), Lugano, Switzerland, March 19-23, 2012*. IEEE Computer Society, 2012, pp. 153–161.

[184] Mehmet Ercan Nergiz, Maurizio Atzori, and Yücel Saygin. “Towards trajectory anonymization: a generalization-based approach”. In: *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS (SPRINGL 2010), Irvine, California, USA, November 4, 2008*. ACM, 2008, pp. 52–61.

[185] Google. *Receiving Location Updates*. Accessed: 2017-03-18. 2013. URL: `https://developer.android.com/training/location/receive-location-updates.html`.

[186] Apple. *Receiving Location Updates*. Accessed: 2017-03-18. 2017. URL: `https://developer.apple.com/library/content/documentation/`

UserExperience / Conceptual / LocationAwarenessPG / CoreLocation/CoreLocation.html.

[187] Mao Ye, Peifeng Yin, and Wang-Chien Lee. “Location recommendation for location-based social networks”. In: *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, November 3-5, 2010, San Jose, CA, USA, Proceedings*. Ed. by Divyakant Agrawal, Pusheng Zhang, Amr El Abbadi, and Mohamed F. Mokbel. ACM, 2010, pp. 458–461. URL: http://doi.acm.org/10.1145/1869790.1869861.

[188] Chris Y. T. Ma, David K. Y. Yau, Nung Kwan Yip, and Nageswara S. V. Rao. “Privacy vulnerability of published anonymous mobility traces”. In: *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking (MOBICOM 2010), Chicago, Illinois, USA, September 20-24, 2010*. ACM, 2010, pp. 185–196.

[189] Kristof Ghys, Bart Kuijpers, Bart Moelans, Walied Othman, Dries Vangoidsenhoven, and Alejandro A. Vaisman. “Map matching and uncertainty: an algorithm and real-world experiments”. In: *17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2009, November 4-6, 2009, Seattle, Washington, USA, Proceedings*. Ed. by Divyakant Agrawal, Walid G. Aref, Chang-Tien Lu, Mohamed F. Mokbel, Peter Scheuermann, Cyrus Shahabi, and Ouri Wolfson. ACM, 2009, pp. 468–471. URL: http://doi.acm.org/10.1145/1653771.1653846.

[190] Steven Fortune. “Voronoi diagrams and Delaunay triangulations”. In: *Handbook of discrete and computational geometry*. CRC Press Ser. Discrete Math. Appl. CRC, Boca Raton, FL, 1997, pp. 377–388.

[191] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhiwen Yu. “Fine-grained Preference-aware Location Search Leveraging Crowdsourced Digital Footprints from LBSNs”. In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp ’13. Zurich, Switzerland: ACM, 2013, pp. 479–488. URL: http://doi.acm.org/10.1145/2493432.2493464.

[192] Jeffrey Hightower and Gaetano Borriello. “Location Systems for Ubiquitous Computing”. In: *IEEE Computer* 34.8 (2001), pp. 57–66.

[193] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek F. Abdelzaher. “Range-free localization schemes for large scale sensor networks”. In: *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MOBICOM 2003), San Diego, CA, USA, September 14-19, 2003*. ACM, 2003, pp. 81–95.

[194] Yiyang Zhao, Yunhao Liu, and Lionel M. Ni. “VIRE: Active RFID-based Localization Using Virtual Reference Elimination”. In: *Proceedings of the 2007 International Conference on Parallel Processing (ICPP 2007), Xi-An, China, September 10-14, 2007*. IEEE Computer Society, 2007, p. 56.

[195] Steinunn Anna Gunnlaugsdttir. *Discover a Museum with BLE App for Indoor Location Video from Eldheimar*. Accessed: 2017-03-18. 2014. URL: `https://locatify.com/blog/discover-a-museum-with-ble-app-for-indoor-location-video-from-eldheimar/`.

[196] *American Airlines undertakes industrys biggest deployment of iBeacons at DFW Airport*. Accessed: 2017-03-18. 2014. URL: `http://www.futuretravelexperience.com/2014/06/american-airlines-undertakes-industrys-biggest-deployment-ibeacons-dfw-airport/`.

[197] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. “Survey of Wireless Indoor Positioning Techniques and Systems”. In: *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 37.6 (2007), pp. 1067–1080.

[198] Yanying Gu, Anthony C. C. Lo, and Ignas G. Niemegeers. “A Survey of Indoor Positioning Systems for Wireless Personal Networks”. In: *IEEE Communications Surveys and Tutorials* 11.1 (2009), pp. 13–32.

[199] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and James Collins. *Global positioning system: theory and practice*. Springer Science & Business Media, 2012.

[200] Nirupama Bulusu, John S. Heidemann, and Deborah Estrin. “GPS-less low-cost outdoor localization for very small devices”. In: *IEEE Personal Communications* 7.5 (2000), pp. 28–34.

[201] Dragos Niculescu and B. R. Badrinath. “Ad Hoc Positioning System (APS) Using AOA”. In: *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, San Franciso, CA, USA, March 30 - April 3, 2003*. IEEE, 2003.

[202] Radhika Nagpal, Howard E. Shrobe, and Jonathan Bachrach. “Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network”. In: *Proceedings of the Information Processing in Sensor Networks (IPSN 2003), Second International Workshop, Palo Alto, CA, USA, April 22-23, 2003*. Vol. 2634. Lecture Notes in Computer Science. Springer, 2003, pp. 333–348.

[203] Leonard Kleinrock and John Silvester. “Optimum transmission radii for packet radio networks or why six is a magic number”. In: *Proceedings of the National Telecommunications Conference (NTC ’78), Birmingham, Ala., December 3-6, 1978*. Vol. 1. Birimingham, Alabama. 1978, pp. 4.3.1–4.3.5.

[204] K. Pahlavan, Xinrong Li, and J.-P. Makela. “Indoor geolocation science and technology”. In: *Communications Magazine, IEEE* 40.2 (2002), pp. 112–118.

[205] Giuseppe Anastasi, Renata Bandelloni, Marco Conti, Franca Delmastro, Enrico Gregori, and Giovanni Mainetto. “Experimenting an Indoor Bluetooth-Based Positioning Service”. In: *Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCS 2003 Workshops), Providence, RI, USA, May 19-22, 2003*. IEEE Computer Society, 2003, pp. 480–483.

[206] Javier J. M. Diaz, Rodrigo de A. Maues, Rodrigo B. Soares, Eduardo Freire Nakamura, and Carlos Mauricio S. Figueiredo. “Bluepass: An indoor Bluetooth-based localization system for mobile applications”. In: *Proceedings of the 15th IEEE Symposium on Computers and Communications (ISCC 2010), Riccione, Italy, June 22-25, 2010*. IEEE Computer Society, 2010, pp. 778–783.

[207] K. Kyamakya, A. Zapater, and Z. Lue. “An indoor Bluetooth-based positioning system: concept, implementation and experimental evaluation”. In: *International* (2003).

[208] Sudarshan S. Chawathe. “Beacon Placement for Indoor Localization using Bluetooth”. In: *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems (ITSC 2008), Beijing, China, October 12-15, 2008*. IEEE, 2008, pp. 980–985.

[209] Paramvir Bahl and Venkata N. Padmanabhan. “RADAR: An In-Building RF-Based User Location and Tracking System”. In: *Proceedings of the IEEE INFOCOM 2000, The Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel Aviv, Israel, March 26-30, 2000*. IEEE, 2000, pp. 775–784.

[210] Paramvir Bahl and Venkata N. Padmanabhan. *Enhancements to the RADAR User Location and Tracking System*. Tech. rep. MSR-TR-2000-12. Microsoft Research, 2000, p. 13.

[211] Veljo Otsason, Alex Varshavsky, Anthony LaMarca, and Eyal de Lara. “Accurate GSM Indoor Localization”. In: *Proceedings of the 7th International Conference on Ubiquitous Computing (UbiComp 2005), Tokyo, Japan, September 11-14, 2005*. Vol. 3660. Lecture Notes in Computer Science. Springer, 2005, pp. 141–158.

[212] Kenneth C. Cheung, Stephen S. Intille, and Kent Larson. “An inexpensive bluetooth-based indoor positioning hack”. In: *Proceedings of the 8th International Conference on Ubiquitous Computing (UbiComp 2006), Orange County, CA, USA, September 17-21, 2006*. Springer-Verlag, 2006.

[213] Sudarshan S. Chawathe. “Low-latency indoor localization using bluetooth beacons”. In: *Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems (ITSC 2009), Missouri USA, October 04-07, 2009*. IEEE, 2009, pp. 1–7.

[214] S. Gezici, Zhi Tian, G.B. Giannakis, Hisashi Kobayashi, A.F. Molisch, H.V. Poor, and Z. Sahinoglu. “Localization via ultra-wideband radios: a look at positioning

aspects for future sensor networks". In: *Signal Processing Magazine, IEEE* 22.4 (2005), pp. 70–84.

[215] Ivan E. Sutherland and G. W. Hodgman. "Reentrant Polygon Clipping". In: *Communications of the ACM* 17.1 (1974), pp. 32–42.

[216] J.C. Liberti and T.S. Rappaport. "A geometrically based model for line-of-sight multipath radio channels". In: *Proceedings of the 1996 IEEE 46th Vehicular Technology Conference (VTC 1996), Atlanta, Georgia, USA, 1996*. Vol. 2. IEEE, 1996, pp. 844–848.

[217] Shahin Farahani. *ZigBee Wireless Networks and Transceivers*. Newton, MA, USA: Newnes, 2011.

[218] Anind K. Dey, Katarzyna Wac, Denzil Ferreira, Kevin Tassini, Jin-Hyuk Hong, and Julian Ramos. "Getting closer: an empirical investigation of the proximity of user to their smart phones". In: *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp 2011), Beijing, China, September 17-21, 2011*. ACM, 2011, pp. 163–172.

[219] Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. López. "Indexing the Positions of Continuously Moving Objects". In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, May 16-18, 2000*. ACM, 2000, pp. 331–342.

[220] Christian S. Jensen, Dan Lin, and Beng Chin Ooi. "Query and Update Efficient B+-Tree Based Indexing of Moving Objects". In: *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB 2004), Toronto, Canada, August 31 - September 3 2004*. Morgan Kaufmann, 2004, pp. 768–779.

[221] Christian S. Jensen, Dalia Tiesyte, and Nerius Tradisauskas. "Robust B+-Tree-Based Indexing of Moving Objects". In: *Proceedings of the 7th International Conference on Mobile Data Management (MDM 2006), Nara, Japan, May 9-13, 2006*. IEEE Computer Society, 2006, p. 12.

[222] Raphael A. Finkel and Jon Louis Bentley. "Quad Trees: A Data Structure for Retrieval on Composite Keys". In: *Acta Informatica* 4.1 (1974), pp. 1–9.

[223] Antonin Guttman. “R-Trees: A Dynamic Index Structure for Spatial Searching”. In: *Proceedings of the SIGMOD’84 Annual Meeting, Boston, Massachusetts, June 18-21, 1984*. ACM Press, 1984, pp. 47–57.

[224] Gísli R. Hjaltason and Hanan Samet. “Ranking in Spatial Databases”. In: *Proceedings of the 4th International Symposium on Advances in Spatial Databases (SSD 1995), Portland, Maine, USA, August 6-9, 1995*. Vol. 951. Lecture Notes in Computer Science. Springer, 1995, pp. 83–95.

[225] J. Kuan and P. Lewis. “Fast $k$ nearest neighbour search for R-tree family”. In: *Proceedings of the 1997 International Conference on Information, Communications and Signal Processing (ICICS 1997), Singapore, September 9-12, 1997*. Institute of Electrical and Electronics Engineers (IEEE), 1997, pp. 924–928.

[226] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[227] Jack A. Orenstein. “Spatial Query Processing in an Object-Oriented Database System”. In: *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 28-30, 1986*. ACM Press, 1986, pp. 326–336.

[228] A Frank. “Problems of realizing LIS: storage methods for space related data: the fieldtree”. In: *Institute for Geodesy and Photogrammetry, Technical Report 71, Zurich, Switzerland, 1983*. Swiss Federal Institute of Technology (EHT), 1983.

[229] Andrew U. Frank and Renato Barrera. “The Fieldtree: A Data Structure for Geographic Information Systems”. In: *Proceedings of the Design and Implementation of Large Spatial Databases (First Symposium SSD’89), Santa Barbara, California, July 17-18, 1989*. Vol. 409. Lecture Notes in Computer Science. Springer, 1989, pp. 29–44.

[230] Hanan Samet, Jagan Sankaranarayanan, and Michael Auerbach. “Indexing methods for moving object databases: games and other applications”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2013), New York, NY, USA, June 22-27, 2013*. ACM, 2013, pp. 169–180.

[231] Donald E. Knuth. *The art of computer programming*. Vol. 3. Sorting and searching, Addison-Wesley Series in Computer Science and Information Processing. Addison Wesley Longman Publishing Co., Inc, 1973, xi+722 pp. (1 foldout).

[232] Xiaohui Yu, Ken Q. Pu, and Nick Koudas. "Monitoring K-Nearest Neighbor Queries Over Moving Objects". In: *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005), Tokyo, Japan, 5-8 April 2005*. IEEE Computer Society, 2005, pp. 631–642.

[233] Yufei Tao, Dimitris Papadias, and Qiongmao Shen. "Continuous Nearest Neighbor Search". In: *Proceedings of 28th International Conference on Very Large Data Bases (VLDB), Hong Kong, China, August 20-23, 2002*. Morgan Kaufmann, 2002, pp. 287–298.

[234] Glenn S. Iwerks, Hanan Samet, and Kenneth P. Smith. "Continuous K-Nearest Neighbor Queries for Continuously Moving Points with Updates". In: *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB 2003), Berlin, Germany, September 9-12, 2003*. 2003, pp. 512–523.

[235] Rimantas Benetis, Christian S. Jensen, Gytis Karciauskas, and Simonas Saltenis. "Nearest and reverse nearest neighbor queries for moving objects". In: *The VLDB Journal* 15.3 (2006), pp. 229–249.

[236] Kenneth L. Clarkson. "Fast Algorithms for the All Nearest Neighbors Problem". In: *Proceedings of the 24th Annual Symposium on Foundations of Computer Science (FOCS 1983), Tucson, Arizona, USA, 7-9 November 1983*. IEEE Computer Society, 1983, pp. 226–232.

[237] Pravin M. Vaidya. "An O(n log n) Algorithm for the All-nearest.Neighbors Problem". In: *Discrete & Computational Geometry* 4 (1989), pp. 101–115.

[238] Jagan Sankaranarayanan, Hanan Samet, and Amitabh Varshney. "A fast all nearest neighbor algorithm for applications involving large point-clouds". In: *Computers & Graphics* 31.2 (2007), pp. 157–174.

[239] Georgios Chatzimilioudis, Demetrios Zeinalipour-Yazti, Wang-Chien Lee, and Marios D. Dikaiakos. "Continuous All k-Nearest-Neighbor Querying in Smartphone Networks". In: *Proceedings of the 13th IEEE International Conference on*

*Mobile Data Management (MDM 2012), Bengaluru, India, July 23-26, 2012*. IEEE Computer Society, 2012, pp. 79–88.

[240] Mordechai Haklay. "How Good is Volunteered Geographical Information? A Comparative Study of OpenStreetMap and Ordnance Survey Datasets". In: *Environment and Planning B: Planning and Design* 37.4 (2010), pp. 682–703.

[241] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones". In: *ACM Transactions on Computer Systems* 32.2 (2014), 5:1–5:29.

[242] Sunny Consolvo, Ian E. Smith, Tara Matthews, Anthony LaMarca, Jason Tabert, and Pauline Powledge. "Location disclosure to social relations: why, when, & what people want to share". In: *Proceedings of the 2005 Conference on Human Factors in Computing Systems (CHI 2005), Portland, Oregon, USA, April 2-7, 2005*. ACM, 2005, pp. 81–90.

[243] Denise Anthony, Tristan Henderson, and David Kotz. "Privacy in Location-Aware Computing Environments". In: *IEEE Pervasive Computing* 6.4 (2007), pp. 64–72.

[244] Eran Toch, Justin Cranshaw, Paul Hankes Drielsma, Jay Springfield, Patrick Gage Kelley, Lorrie Faith Cranor, Jason I. Hong, and Norman M. Sadeh. "Locaccino: a privacy-centric location sharing application". In: *Proceedings of the 12th International Conference of Ubiquitous Computing (UbiComp 2010), Adjunct Papers Proceedings, Copenhagen, Denmark, September 26-29, 2019*. ACM International Conference Proceeding Series. ACM, 2010, pp. 381–382.

[245] Jason Wiese, Patrick Gage Kelley, Lorrie Faith Cranor, Laura Dabbish, Jason I. Hong, and John Zimmerman. "Are you close with me? are you nearby?: investigating social groups, closeness, and willingness to share". In: *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp 2011), Beijing, China, September 17-21, 2011*. ACM, 2011, pp. 197–206.

[246] Daniel T. Wagner, Andrew Colin Rice, and Alastair R. Beresford. "Device Analyzer: Understanding Smartphone Usage". In: *Proceedings of the 10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Ser-*

*vices (MOBIQUITOUS 2013), Tokyo, Japan, December 2-4, 2013*. Vol. 131. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, 2013, pp. 195–208.

[247] Noam Ben-Asher, Niklas Kirschnick, Hanul Sieger, Joachim Meyer, Asaf Ben-Oved, and Sebastian Möller. "On the need for different security methods on mobile phones". In: *Proceedings of the 13th Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI 2011), Stockholm, Sweden, August 30 - September 2, 2011*. ACM, 2011, pp. 465–473.

[248] Peter Eckersley. "How Unique Is Your Web Browser?" In: *Proceedings of the 10th International Conference on Privacy Enhancing Technologies (PETS 2010), Berlin, Germany, July 21-23, 2010*. Vol. 6205. Lecture Notes in Computer Science. Springer Nature, 2010, pp. 1–18.

[249] Andrzej Zolnierek and Bartlomiej Rubacha. "The Empirical Study of the Naive Bayes Classifier in the Case of Markov Chain Recognition Task". In: *Proceedings of the 4th International Conference on Computer Recognition Systems (CORES'05), Rydzyna Castle, Poland, May 22-25, 2005*. Vol. 30. Advances in Soft Computing. Springer, 2005, pp. 329–336.

[250] Adrian Holzer and Jan Ondrus. "Trends in Mobile Application Development". In: *Proceedings of the Second International Conference on Mobile Wireless Middleware, Operating Systems, and Applications - Workshops (Mobilware 2009 Workshops), Berlin, Germany, April 28-29, 2009*. Vol. 12. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, 2009, pp. 55–64.

[251] Zarni Aung and Win Zaw. "Permission-based android malware detection". In: *International Journal of Scientific and Technology Research* 2.3 (2013), pp. 228–234.

[252] Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo García Bringas, and Gonzalo Álvarez Marañón. "PUMA: Permission Usage to Detect Malware in Android". In: *Proceedings of the International Joint Conference CISIS'12-ICEUTE'12-SOCO'12 Special Sessions, Ostrava, Czech Republic, September 5th-7th, 2012*. Vol. 189. Advances in Intelligent Systems and Computing. Springer, 2012, pp. 289–298.

[253] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. "Crowdroid: behavior-based malware detection system for Android". In: *Proceedings of the 1st ACM Workshop Security and Privacy in Smartphones and Mobile Devices (SPSM 2011), Co-located with CCS 2011, Chicago, IL, USA, October 17, 2011*. ACM, 2011, pp. 15–26.

[254] Bruno R. Preiss. *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*. Wiley, 1999.

[255] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, and David Lie. "PScout: analyzing the Android permission specification". In: *Proceedings of the 2012 ACM conference on Computer and Communications Security (CCS 2012), Raleigh, NC, USA October 16 - 18, 2012*. ACM, 2012, pp. 217–228.

[256] Hiroshi Lockheimer. *Android and Security*. Accessed: 2016-09-01. 2012. URL: `http://googlemobile.blogspot.com/2012/02/android-and-security.html`.

[257] Jon Oberheide and Charlie Miller. *Dissecting the Android Bouncer*. SummerCon2012, New York: https://jon.oberheide.org/files/summercon12-bouncer.pdf. Accessed: 2016-09-01. 2012.

[258] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, Phillipa Gill, and David Lie. "Short paper: a look at smartphone permission models". In: *Proceedings of the 1st ACM Workshop Security and Privacy in Smartphones and Mobile Devices, Co-located with CCS 2011, Chicago, IL, USA, October 17, 2011*. ACM, 2011, pp. 63–68.

[259] David Barrera, Hilmi Günes Kayacik, Paul C. van Oorschot, and Anil Somayaji. "A methodology for empirical analysis of permission-based security models and its application to android". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2010), Chicago, Illinois, USA, October 4-8, 2010*. ACM, 2010, pp. 73–84.

[260] Jialiu Lin, Bin Liu, Norman M. Sadeh, and Jason I. Hong. "Modeling Users' Mobile App Privacy Preferences: Restoring Usability in a Sea of Permission Settings". In: *Proceedings of the Tenth Symposium on Usable Privacy and Security*

*(SOUPS 2014), Menlo Park, CA, USA, July 9-11, 2014*. USENIX Association, 2014, pp. 199–212.

[261] Bhaskar Pratim Sarma, Ninghui Li, Christopher S. Gates, Rahul Potharaju, Cristina Nita-Rotaru, and Ian Molloy. “Android permissions: a perspective combining risks and benefits”. In: *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies (SACMAT ’12), Newark, NJ, USA - June 20 - 22, 2012*. ACM, 2012, pp. 13–22.

[262] William Enck, Machigar Ongtang, and Patrick Drew McDaniel. “On lightweight mobile phone application certification”. In: *Proceedings of the 2009 ACM Conference on Computer and Communications Security (CCS 2009), Chicago, Illinois, USA, November 9-13, 2009*. ACM, 2009, pp. 235–245.